Al-Azhar University

Faculty of Engineering

Electrical Engineering Department

(Electronics & Communication)

B.Sc.

GRADUATION PROJECT

Application On Ad Hoc Using Wireless Sensor Networks

Supervised by :

Prof. Abdellatif I. Moustafa

PRESENTED BY:

Omar Mohamed Mohamed Abdelhameid

Ahmed Mohamed Taha Mohamed

Abdelrahman Shaban Sayed Sayed

Abdelrahman Ali Eid Sayed

Ali Ibrahim Mohamed Ali

Mahmoud Mohamed Elsayed

July 2026

Acknowledgment

In the name of Allah, the Most Gracious, the Most Merciful.

All praise and gratitude are due to Allah, whose countless blessings, guidance, and mercy have enabled us to complete this work. We are deeply thankful for the strength, patience, and perseverance He bestowed upon us throughout this journey.

We extend our deepest appreciation to our esteemed supervisor, **Prof. Dr. Abdel Latif Ibrahim Mostafa**, for his invaluable mentorship, unwavering support, and insightful guidance throughout the development of this project. His constructive feedback, profound expertise, and commitment to academic excellence continuously inspired us to refine our work and pursue the highest standards of quality. His encouragement and confidence in our abilities have been instrumental in bringing this project to fruition.

We are equally grateful to the distinguished faculty members of the **Department of Communications and Electronics Engineering**, **Faculty of Engineering, Al-Azhar University**, whose dedication to teaching and commitment to excellence have provided us with the knowledge, skills, and intellectual foundation essential to our academic and professional growth. Their guidance throughout our years of study has played a fundamental role in preparing us for this achievement.

Our sincere appreciation also goes to every member of our project team. The spirit of collaboration, mutual respect, and shared responsibility that characterized our teamwork transformed challenges into opportunities for learning and innovation. This project stands as a reflection of our collective effort, commitment, and determination.

Finally, we express our heartfelt gratitude to our families for their unconditional love, endless encouragement, and steadfast support. Their faith in our potential and their continuous motivation have been a constant source of strength throughout our academic journey.

To everyone who contributed, directly or indirectly, to the successful completion of this project, we offer our sincere appreciation and profound thanks. We hope that this work will serve as a meaningful contribution to knowledge and inspire future endeavors in the field.

Preface

Wireless Sensor Networks (WSNs) have become one of the most effective technologies for monitoring and controlling distributed environments. Their capability to collect, transmit, and process data in real time has enabled numerous applications in areas such as environmental monitoring, industrial automation, healthcare, and smart cities. Among these applications, forest fire detection represents a critical challenge where early identification of hazardous conditions can significantly reduce environmental, economic, and human losses.

Motivated by this challenge, this book presents the design and implementation of a **WSN-based forest fire detection system** as a practical application of wireless sensing and intelligent monitoring technologies. The proposed system continuously monitors key environmental parameters, processes the acquired data to determine the current fire risk level, and provides real-time cloud monitoring and automated warning notifications whenever dangerous conditions are detected.

To provide a clear understanding of the proposed system, the book is organized in a progressive manner. **Chapter 1** introduces the fundamentals of computer networks, providing the communication concepts required for wireless sensing systems. Building upon this foundation, **Chapter 2** presents the principles, architecture, and characteristics of Wireless Sensor Networks and discusses their role in environmental monitoring applications. **Chapter 3** introduces the LabVIEW development environment and explains its integration with the selected WSN gateway for real-time data acquisition.

After establishing the theoretical and technical background, **Chapter 4** formulates the forest fire detection problem by identifying the monitored environmental parameters, defining the five fire risk levels, and presenting the methodology adopted for risk assessment based on real environmental thresholds. **Chapter 5** describes the practical implementation of the proposed system, including sensor data acquisition using LabVIEW, TCP communication with Python, data processing and fire risk classification, Firebase cloud integration, and automated alert generation. Finally, **Chapter 6** concludes the work by summarizing the project's main achievements and presenting recommendations for future improvements.

It is hoped that this book will provide a practical reference for students, researchers, and engineers interested in Wireless Sensor Networks and intelligent environmental monitoring systems, while demonstrating how modern sensing, communication, and cloud technologies can be integrated into a complete real-time forest fire detection solution.

Table of Contents

4

CHAPTER 1: INTRODUCTION TO NETWORKS 3
1.1 Overview 4
1.2 Network Definition 4
1.2.1 Definition of a Network 4
1.2.2 Objectives of Networks 4
1.2.3 Advantages of Networks 5
1.3 Types of Networks 5
1.3.1 Computer Networks 6
1.3.2 Communication Networks 6
1.3.3 Wireless Networks 6
1.3.4 Wireless Sensor Networks 6
1.4 Communication Systems 8
1.4.1 Communication System Components 8
1.4.2 Wired versus Wireless Communication 9
1.5 Network Topologies 10
1.6 Wireless Sensor Networks 12
1.6.1 Definition 12
1.6.2 Architecture 12
1.6.3 Components 13
1.6.4 Characteristics 14
1.7 Ad Hoc Networks 15
1.7.1 Definition 15
1.7.2 Features 15
1.7.3 Types of Ad Hoc Networks 16
1.7.4 Advantages 16
1.7.5 Challenges 16
1.8 Relationship Between WSN and Ad Hoc Networks 17
1.8.1 WSN as a Subclass of Ad Hoc Networks 18
1.8.2 Point-by-Point Comparison 18
1.8.3 How This Project's Network Relates to Ad Hoc Principles 19
1.9 Applications of WSN 21
1.10 Chapter Summary 22
CHAPTER 2: WIRELESS SENSOR NETWORKS (WSN) 23
2.1 Introduction 24
2.2 Wireless Sensor Networks (WSNs) 24

2.3 Components of Wireless Sensor Networks 25
2.3.1 Sensor Nodes 26
2.3.2 Gateway 27
2.3.3 Communication Network 27
2.3.4 Monitoring Station 27
2.4 Wireless Sensor Network Architecture 28
2.4.1 Sensing Layer 28
2.4.2 Communication Layer 29
2.4.3 Processing Layer 29
2.4.4 Application Layer 30
2.5 Characteristics of Wireless Sensor Networks 30
2.5.1 Low Power Consumption 30
2.5.2 Wireless Connectivity 30
2.5.3 Scalability 31
2.5.4 Reliability 31
2.5.5 Self-Organization 31
2.5.6 Flexibility 32
2.6 National Instruments Wireless Sensor Network Platform 33
2.7 NI WSN-3202 Analog Input Node 34
2.7.1 Introduction 34
2.7.2 Hardware Architecture 35
2.7.3 Analog Input Characteristics 37
2.7.4 Input Voltage Ranges 37
2.7.5 Measurement Accuracy 38
2.7.6 Wireless Communication Characteristics 38
2.7.7 Communication Range 39
2.7.8 Antenna System 39
2.7.9 Power Supply Options 39
2.7.10 Power Consumption 40
2.7.11 Battery Lifetime 40
2.7.12 Digital Input and Output Features 41
2.7.13 Environmental Specifications 41
2.7.14 Physical Characteristics 42
2.7.15 Advantages of NI WSN-3202 42
2.8 NI WSN-3212 Thermocouple Node 43
2.8.1 Introduction 43
2.8.2 Device Overview 44
2.8.3 Hardware Architecture 44
2.8.4 Thermocouple Measurement Capability 46

2.8.5 High-Resolution Data Acquisition 46
2.8.6 Wireless Communication System 47
2.8.7 Communication Range 47
2.8.8 Power Supply Options 47
2.8.9 Low-Power Operation 48
2.8.10 Battery Lifetime 48
2.8.11 Environmental Specifications 48
2.8.12 Mechanical Characteristics 49
2.8.13 Network Integration 49
2.8.14 Advantages of NI WSN-3212 50
2.9 NI WSN-9791 Ethernet Gateway 51
2.9.1 Introduction 51
2.9.2 Gateway Overview 52
2.9.3 Hardware Architecture 52
2.9.4 Network Coordination Functions 53
2.9.5 Wireless Communication Technology 54
2.9.6 Ethernet Connectivity 54
2.9.7 Network Capacity 54
2.9.8 Node Registration Process 55
2.9.9 Data Flow within the Network 55
2.9.10 Status Monitoring and Diagnostics 56
2.9.11 Reliability and Industrial Performance 56
2.9.12 Security Considerations 57
2.9.13 Advantages of the NI WSN-9791 Gateway 57
2.9.14 Applications of the NI WSN-9791 Gateway 57
2.10 Wireless Sensor Network Topology 58
2.10.1 Introduction 58
2.10.2 Star Topology 58
2.10.3 Operation of Star Networks 59
2.10.4 Advantages of Star Topology 59
2.10.5 Limitations of Star Topology 60
2.11 Power Management in Wireless Sensor Networks 60
2.11.1 Introduction 60
2.11.2 Energy Consumption Sources 60
2.11.3 Sleep Mode Operation 61
2.11.4 Battery Lifetime Considerations 62
2.11.5 Importance of Energy Efficiency 63
2.12 Wireless Sensor Network Performance Metrics 63
2.12.1 Introduction 63

2.12.2 Network Lifetime 63
2.12.3 Communication Reliability 64
2.12.4 Scalability 64
2.12.5 Latency 65
2.12.6 Throughput 66
2.13 Advantages of Wireless Sensor Networks in Environmental Monitoring 66
2.13.1 Introduction 66
2.13.2 Reduced Installation Cost 67
2.13.3 Flexible Deployment 67
2.13.4 Real-Time Monitoring 68
2.13.5 Remote Accessibility 69
2.13.6 Scalability and Expandability 69
2.13.7 Energy Efficiency 69
2.13.8 Reliability 69
2.14 Communications Protocols 69
2.14.1 The Protocol Stack 69
2.14.2 Physical Layer 70
2.14.3 MAC Protocols 71
2.14.4 Routing Protocols 72
2.14.5 Transport and Application Layers 72
2.15 Infrastructure-less Wireless Sensor Networks 73
2.16 Chapter Summary 74
CHAPTER 3: INTRODUCTION TO LABVIEW 75
3.1 Introduction 76
3.2 Overview of LabVIEW 76
3.2.1 History and Development 76
3.2.2 LabVIEW Programming Model 77
3.3 LabVIEW Key Components and Features 78
3.3.1 Virtual Instruments (VIs) 78
3.3.2 Data Types and Wires 78
3.3.3 Programming Structures 78
3.3.4 Data Acquisition (DAQ) Subsystem 79
3.4 LabVIEW in Wireless Sensor Networks 80
3.4.1 Role of LabVIEW in WSN Applications 80
3.4.2 Communication Interfaces 81
3.5 Data Acquisition and Processing in LabVIEW 82
3.5.1 Data Acquisition Pipeline 82
3.5.2 Signal Processing 83
3.5.3 Data Logging and File I/O 83

3.6 Introduction to Virtual Instruments and the LabVIEW Environment 85
3.7 Building Front Panel and Block Diagram 88
3.7.1 The Front Panel 88
3.7.2 The Block Diagram 88
3.8 Running and Debugging VIs 90
3.8.1 Running a VI 90
3.8.2 Debugging Techniques 91
3.9 Creating VIs and SubVIs 92
3.9.1 Creating a Virtual Instrument (VI) 92
3.9.2 Creating and Using SubVIs 92
3.10 Structures and Loops 93
3.11 Grouping Data 95
3.11.1 Grouping Data with Strings 95
3.11.2 Grouping Data with Arrays 96
3.11.3 Grouping Data with Clusters 97
3.12 Graphs and Charts 97
3.12.1 Types of Graphs and Charts 97
3.12.2 Customizing Graphs and Charts 98
3.13 File I/O 99
3.13.1 Core Concepts 99
3.13.2 File Types and Data Handling 99
3.14 Documenting and Printing 100
3.14.1 To add a description to a VI, open the VI Properties dialog and navigate to the 100
3.14.2 Printing VIs: LabVIEW allows you to print the Front Panel, Block Diagram, and 101
3.15 Connecting the NI WSN-9791 Gateway to LabVIEW 103
3.15.1 Prerequisites 103
3.15.2 Gateway Configuration 103
3.15.3 LabVIEW Integration 106
3.16 Chapter Summary 107
CHAPTER 4: PROJECT PROBLEM FORMULATION 108
4.1 Introduction 109
4.2 Forest Fire Problem Analysis 110
4.3 Existing Fire Detection Methods 112
4.4 Problem Statement 114
4.5 System Requirements 116
4.5.1 Functional Requirements 116
4.5.2 Non-Functional Requirements 118

4.6 Proposed Solution 119
4.7 System Architecture 121
4.8 Soil Moisture Sensor 123
4.8.1 Introduction 123
4.8.2 Operating Principle 123
4.8.3 Technical Specifications 124
4.8.4 Hardware Integration 124
4.8.5 Calibration and Data Conversion 124
4.8.6 Moisture-Based Risk Classification 125
4.8.7 Role in Fire Detection System 125
4.9 MQ-135 Air Quality and Carbon Dioxide Sensor 126
4.9.1 Introduction 126
4.9.2 Operating Principle 126
4.9.3 Technical Specifications 127
4.9.4 Hardware Integration 127
4.9.5 Calibration and CO_2 Concentration Calculation 128
4.9.6 Fire Detection Threshold 128
4.9.7 Role in Fire Detection System 128
4.10 MQ-7 Carbon Monoxide Sensor 129
4.10.1 Introduction 129
4.10.2 Operating Principle 130
4.10.3 Technical Specifications 130
4.10.4 Hardware Integration 130
4.10.5 Calibration and CO Concentration Calculation 131
4.10.6 Smoldering Combustion Classification 132
4.10.7 Role in Fire Detection Algorithm 132
4.10.8 Advantages of Using MQ-7 in Forest Fire Detection 133
4.11 K-Type Thermocouple Temperature Sensor 133
4.11.1 Introduction 133
4.11.2 Operating Principle 134
4.11.3 Technical Specifications 134
4.11.4 Hardware Integration 135
4.11.5 Environmental Considerations 135
4.11.6 Temperature Threshold Classification 136
4.11.7 Role in Fire Detection Algorithm 136
4.11.8 Advantages of Using K-Type Thermocouple 136
4.12 Fire Detection Algorithm 137
4.12.1 Introduction 137
4.12.2 Data Processing Sequence 138

4.12.3 Sensor Validation 138
4.12.4 Multi-Parameter Decision Logic 139
4.12.5 Decision Strategy 139
4.12.6 Advantages of the Proposed Algorithm 140
4.12.7 Role within the Proposed System 140
4.13 Fire Risk Classification 141
4.13.1 Introduction 141
4.13.2 Classification Levels 142
4.13.3 Level 0 – NORMAL 142
4.13.4 Level 1 – LOW WARNING 142
4.13.5 Level 2 – MEDIUM DANGER 143
4.13.6 Level 3 – HIGH DANGER 143
4.13.7 Level 4 – CRITICAL 143
4.13.8 Notification Response According to Danger Level 144
4.13.9 Benefits of Multi-Level Classification 144
4.14 Cloud Monitoring and Notification System 145
4.14.1 Introduction 145
4.14.2 TCP Communication between LabVIEW and Python 145
4.14.3 Firebase Realtime Database 145
4.14.4 E-Mail Notification System 146
4.14.5 Automated Voice Call System 146
4.14.6 Notification Strategy 147
4.14.7 Advantages of Cloud-Based Monitoring 147
4.15 Expected Outcomes 148
4.16 Chapter Summary 149
CHAPTER 5: PROJECT IMPLEMENTATION 150
5.1 Introduction 151
5.2 Preliminary Architecture — Common Sampling Rate Design 152
5.2.1 System Architecture Overview 152
5.2.2 Queue Communication Architecture 152
5.2.3 Common Sampling Rate Implementation 154
5.3 LabVIEW Virtual Instrument Implementation 155
5.3.1 Soil Moisture Sensor VI 155
5.3.2 CO_2 Sensor VI (MQ-135) 157
5.3.3 CO Sensor VI (MQ-7) 159
5.3.4 Temperature Sensor VI (Dual Thermocouple) 161
5.3.5 Main Control Loop VI 163
5.4 TCP Communication — LabVIEW Client to Python Server 167
5.4.1 Connection Parameters 167

5.4.2 Communication Sequence 167
5.5 Python Backend Implementation 169
5.5.1 Software Architecture and Module Dependencies 169
5.5.2 Sensor Validation and Sanitization 169
5.5.3 Danger Classification Algorithm — classify_sensors() 170
5.5.4 Firebase Realtime Database Integration 173
5.5.5 Alert Notification System 174
5.6 Enhanced Final System Architecture 176
5.6.1 Architectural Improvements Overview 176
5.6.2 Notifier Architecture 176
5.6.3 Independent Sensor Loop Design 177
5.6.4 Data Flow Architecture 178
5.7 Comparative Analysis — Preliminary vs. Enhanced Architecture 179
5.7.1 Engineering Discussion 179
5.8 Experimental Results and Sensor Measurements 181
5.9 Power Consumption and Battery Life Analysis 191
5.9.1 System Overview 191
5.9.2 Node Current Consumption and Battery Life 191
5.9.3 Key Equations 191
5.9.4 Fixed vs. Adaptive Sampling — Comparative Analysis 192
5.9.5 Battery Life Results Summary 192
5.9.6 Adaptive Sampling Decision Flowchart 192
5.9.7 Discussion 192
5.10 Wireless Communication Range Test 193
5.11 Chapter Summary 194
CHAPTER 6: CONCLUSION AND FUTURE WORK 195
6.1 Conclusion 195
6.2 Future Work 197

List of figures

Figure 1.1: Classification of networks 5
Figure 1.2: Types of networks 7
Figure 1.3: Generic communication system block diagram 8
Figure 1.4: Wired versus wireless communication 9
Figure 1.5: Common network topologies 10
Figure 1.6: System-level architecture of a WSN deployment 13
Figure 1.7: Functional block diagram of a wireless sensor node 14
Figure 1.8: Direct communication failing beyond radio range 15
Figure 1.9: WSN as a specialized subclass of wireless ad hoc networks 18
Figure 1.10: Actual star/single-hop topology versus pure ad hoc multi-hop topology 20
Figure 2.1: General Architecture of a Wireless Sensor Network 25
Figure 2.2: Basic Components of a Wireless Sensor Node 26
Figure 2.3: Wireless Sensor Network Layered Architecture 28
Figure 2.4: Environmental Monitoring Using Wireless Sensor Network 32
Figure 2.5: Star Topology in Wireless Sensor Networks 33
Figure 2.6: WSN-3202 Analog Input Node 34
Figure 2.7: NI WSN-3202 Device Dimensions 35
Figure 2.8: NI WSN-3202 Node Interface 36
Figure 2.9: NI WSN-3212 Thermocouple Node 43
Figure 2.10: NI WSN-3212 Device Dimension 44
Figure 2.11: NI WSN-3212 Node Interface 45
Figure 2.12: NI WSN-9791 Ethernet Gateway 51
Figure 2.13: WSN-9791 Ethernet Gateway Dimension 52
Figure 2.14: Star Topology of the NI Wireless Sensor Network 58
Figure 2.15: Main Sources of Energy Consumption in a Wireless Sensor Node 61
Figure 2.16: Sleep–Wake Operation Cycle of a Wireless Sensor Node 62
Figure 2.17: Effect of Sampling Interval on Battery Lifetime 62
Figure 2.18: Reliable Wireless Communication between Sensor Node and Gateway 64
Figure 2.19: Scalability of a Wireless Sensor Network 65
Figure 2.20: Data Transmission Latency in a Wireless Sensor Network 66
Figure 2.21: Wireless Sensor Network for Environmental Monitoring 67
Figure 2.22: Flexible Deployment of Wireless Sensor Nodes 68
Figure 2.23: Real-Time Monitoring Using LabVIEW 68
Figure 2.24: WSN Protocol Stack 70
Figure 2.25: IEEE 802.15.4 Physical Layer 70

Figure 2.26: TDMA vs. CSMA/CA MAC Protocols 71
Figure 2.27: WirelessHART and ISA-100.11a Based on IEEE 802.15.4 72
Figure 2.28: Routing Communication Between Sensor Nodes and the NI WSN Gateway 72
Figure 2.29:ata Flow from Sensor Nodes to LabVIEW 73
Figure 2.30: Infrastructure-less Wireless Sensor Network for Forest Monitoring 73
Figure 3.1: Front Panel (left) and Block Diagram (right) of a Virtual Instrument (VI)..... 77
Figure 3.2: LabVIEW 2013 startup interface. 85
Figure 3.3: Controls Palette: Only available on the Front Panel. 86
Figure 3.4: Functions Palette: Only available on the Block Diagram. 86
Figure 3.5: The Tools Palette. 87
Figure 3.6: The toolbar and execution controls. 87
Figure 3.7: Building the Front Panel..... 88
Figure 3.8: The Block Diagram. 89
Figure 3.9: Run Button and Broken Run Arrow. 90
Figure 3.10: Run Continuously. 90
Figure 3.11: Abort Execution. 90
Figure 3.12: Pause Button. 90
Figure 3.13: Highlight Execution Button..... 91
Figure 3.14: Debugging Tools. 91
Figure 3.15: Connector Pane..... 92
Figure 3.16: VI Icon (as it appears on the Block Diagram). 92
Figure 3.17: SubVI (used in the Block Diagram). 93
Figure 3.18: Structures and Loops. 93
Figure 3.19: Grouping Data with Strings. 95
Figure 3.20: Grouping Data with Arrays. 96
Figure 3.21: Grouping Data with Clusters. 97
Figure 3.22: Graphs and Charts. 98
Figure 3.23: File I/O 99
Figure 3.24: File Menu Showing VI Properties Option. 100
Figure 3.25: VI Properties Dialog – Category Selection 100
Figure 3.26: VI Properties Dialog – Documentation Category. 101
Figure 3.27: File Menu Showing Print Option. 101
Figure 3.28: Print Dialog – Select VI and Print Contents. 102
Figure 3.29: Saving the Print Output as PDF. 102
Figure 3.30: Network Connection Settings 104
Figure 3.31: IPv4 Configuration for Communication with the NI WSN-9791 Gateway 104
Figure 3.32: NI MAX System Settings for the Host Computer. 105
Figure 3.33: NI WSN-9791 Gateway System Settings in NI MAX 105
Figure 3.34: WSN Node Configuration in NI MAX. 106

Figure 4.1: General architecture of a Wireless Sensor Network (WSN)-based forest fire monitoring system. 110
Figure 4.2: Aerial view of a California wildfire illustrating the rapid spread and large-scale environmental impact of forest fires. 111
Figure 4.3: Traditional forest fire monitoring using observation towers. 113
Figure 4.4: Concept of early forest fire detection using environmental monitoring systems. 115
Figure 4.5: General requirements of a real-time environmental monitoring system including sensing, wireless communication, data processing, cloud connectivity, and alert generation. 116
Figure 4.6: General architecture of the proposed Wireless Sensor Network-based forest fire detection system. 120
Figure 4.7: Architecture of a Wireless Sensor Network-based environmental monitoring system. 122
Figure 4.8: Soil Moisture Sensor used for measuring the water content of soil. 123
Figure 4.9: MQ-135 gas sensor used for air quality monitoring and carbon dioxide detection. 126
Figure 4.10: MQ-7 carbon monoxide gas sensor used for combustion gas detection. 129
Figure 4.11: K-Type thermocouple temperature sensor used for environmental temperature monitoring. 133
Figure 4.12: General flowchart of the fire detection and danger classification algorithm. ... 137
Figure 4.13: Multi-level fire risk assessment and danger classification concept. 141
Figure 5.1: Full system integration. 151
Figure 5.2: Experimental Verification of TCP Communication between LabVIEW and Python. 153
Figure 5.3: Full preliminary architecture LabVIEW code. 154
Figure 5.4: Soil Moisture Sensor VI — LabVIEW Block Diagram 156
Figure 5.5: Soil Moisture Sensor VI — Processing Flowchart. 156
Figure 5.6: CO_2 Sensor VI (MQ-135) — LabVIEW Block Diagram 158
Figure 5.7: CO_2 Sensor VI — Processing Flowchart 158
Figure 5.8: CO Sensor VI (MQ-7) — LabVIEW Block Diagram 160
Figure 5.9: CO Sensor VI — Processing Flowchart 160
Figure 5.10: Temperature Sensor VI — LabVIEW Block Diagram 162
Figure 5.11: Temperature Sensor VI — Processing Flowchart 162
Figure 5.12: Main Control Loop VI — LabVIEW Block Diagram 165
Figure 5.13: Main Control Loop — Processing Flowchart 166
Figure 5.14: TCP Communication VI — LabVIEW Block Diagram 168
Figure 5.15: TCP Communication Sequence Flowchart 168
Figure 5.16: classify_sensors() — Danger Classification Flowchart 171
Figure 5.17: Python Main Program — Complete Execution Flowchart 172

Figure 5.18: Firebase Upload — Processing Flowchart 174
Figure 5.19: Email and Phone Alert System — Flowchart 175
Figure 5.20: Safe state – LabVIEW 181
Figure 5.21: Safe state – Firebase 181
Figure 5.22: Low Warning state – LabVIEW 182
Figure 5.23: Low Warning state – Firebase 182
Figure 5.24: Low Warning state – Mail 183
Figure 5.25: Medium Warning state – LabVIEW 183
Figure 5.26: Medium Warning state – Firebase 184
Figure 5.27: Medium Warning state – Mail 184
Figure 5.28: Medium Warning state – call 185
Figure 5.29: High Warning state – LabVIEW 185
Figure 5.30: High Warning state – Firebase 186
Figure 5.31: High Warning state – Mail 186
Figure 5.32: High Warning state – call 187
Figure 5.33: Critical state – LabVIEW 187
Figure 5.34: Critical state – Firebase 188
Figure 5.35: Critical state – Mail 188
Figure 5.36: Critical state – call 189
Figure 5.37: Connection loss state – Mail 189
Figure 5.38: Excel-Based Sensor Data Logging 190
Figure 5.39: Adaptive Sampling Decision and Energy Flow. 192

List of tables

Table 1.1: Comparison of wired and wireless communication 9
Table 1.2: Infrastructure-based networks versus ad hoc networks 17
Table 1.3: WSN versus classical Ad Hoc networks 19
Table 2.1: WSN-3202 Analog Input Specifications 37
Table 2.2: WSN-3202 Analog Input Measurement Accuracy 38
Table 2.3: WSN-3202 Wireless Communication Specifications 38
Table 2.4: WSN-3202 Wireless Range 39
Table 2.5: WSN-3202 Antenna Characteristics 39
Table 2.6: WSN-3202 Power Consumption under Battery Operation 40
Table 2.7: WSN-3202 Power Consumption under External Power Operation 40
Table 2.8: Expected Battery Lifetime of the WSN-3202 41
Table 2.9: WSN-3202 Environmental Specification 41
Table 2.10: WSN-3202 Physical Characteristics 42
Table 2.11: WSN-3212 Thermocouple Measurement Characteristics 46
Table 2.12: WSN-3212 Wireless Communication Specifications 47
Table 2.13: WSN-3212 Wireless Communication Range 47
Table 2.14: WSN-3212 Environmental Specifications 49
Table 2.15: WSN-9791 Wireless Communication Specification 54
Table 2.16: WSN-9791 Network Capacity Specifications 55
Table 4.1: Comparison of Forest Fire Detection Methods 112
Table 4.2: Technical Specifications (Soil Moisture Sensor) 124
Table 4.3: Moisture-Based Risk status 125
Table 4.4: Technical Specifications (MQ-135 Sensor) 127
Table 4.5: Technical specifications of the MQ-7 sensor. 130
Table 4.6: Classification criteria implemented in the proposed system. 132
Table 4.7: Specifications of the K-Type Thermocouple used in the proposed system. 134
Table 4.8: Implemented temperature classification criteria. 136
Table 4.9: Fire Risk Classification Levels 142
Table 4.10: Actions taken by the system for each danger level. 144
Table 4.11: Notification Escalation Strategy 147
Table 5.1: LabVIEW Queue Operation Summary. 153
Table 5.2: Preliminary Architecture — Common Sampling Rate 154
Table 5.3: Soil Moisture Classification Thresholds 155
Table 5.4: CO Concentration Classification Thresholds 159
Table 5.5: Temperature Sensor Channel Configuration 161

Table 5.6: TCP Transmission Packet — Field Definitions 163
Table 5.7: Dynamic Sampling Rate Table — Based on Danger Level 164
Table 5.8: TCP Connection Configuration Parameters 167
Table 5.9: Python Backend Module Dependencies 169
Table 5.10: Boolean Classification Flags — Definition and Thresholds 170
Table 5.11: Danger Classification Decision Table 170
Table 5.12: Firebase /sensors Node — Field Definitions 173
Table 5.13: LabVIEW Notifier Operation Summary 177
Table 5.14: Architecture Comparison — Preliminary vs. Enhanced Final Design 179
Table 5.15: Node current consumption summary. (* = estimated) 191
Table 5.16: Fixed vs. Adaptive Sampling strategy comparison. 192

List of Abbreviation

Abbreviation	Definition
ADC	Analog-to-Digital Converter
AI	Analog Input
AO	Analog Output
CJC	Cold Junction Compensation
CO	Carbon Monoxide
CO_2	Carbon Dioxide
CRC	Cyclic Redundancy Check
DAQ	Data Acquisition
DIO	Digital Input/Output
DSSS	Direct Sequence Spread Spectrum
FFT	Fast Fourier Transform
FIFO	First In, First Out
GUI	Graphical User Interface
IoT	Internet of Things
ISM	Industrial, Scientific, and Medical
LAN	Local Area Network
LQ	Link Quality
LQI	Link-Quality Index
MAN	Metropolitan Area Network
MANET	Mobile Ad Hoc Network
ML	Machine Learning
MOS	Metal-Oxide Semiconductor
NI	National Instruments
PMM	Piece-wise Monotonic Mapping
RMS	Root Mean Square
SnO_2	Tin Dioxide
TCP	Transmission Control Protocol
TCP/IPv4	Internet Protocol version 4
TDMA	Time Division Multiple Access
TSCH	Time-Slotted Channel Hopping
UAV	Drone
UDP	User Datagram Protocol

VANET	Vehicular Ad Hoc Network
VI	Virtual Instrument
VOCs	Volatile Organic Compounds
WAN	Wide Area Network
WSN	Wireless Sensor Network

CHAPTER

1

Introduction to Networks

Network Definition

Types of Networks

Communication Systems

Network Topologies

Wireless Sensor Networks

WSN & Ad Hoc Networks

1.1 Overview

Networks form the backbone of virtually every modern technological system, enabling devices, machines, and people to exchange information regardless of physical distance. From global telecommunication infrastructures to small-scale embedded systems, the ability to connect, communicate, and share data has become a fundamental requirement of modern engineering design. In recent years, the demand for distributed, autonomous, and intelligent monitoring systems has driven significant interest in Wireless Sensor Networks (WSNs), which combine sensing, processing, and wireless communication into compact, low-power devices capable of operating in remote or hostile environments. This chapter introduces the fundamental concepts of networks in general, before narrowing the focus to wireless networks, Wireless Sensor Networks, and Ad Hoc networks, which together form the theoretical foundation of this graduation project.

1.2 Network Definition

1.2.1 Definition of a Network

A network is broadly defined as an interconnected group of two or more devices, nodes, or systems that are linked together for the purpose of exchanging data, sharing resources, or coordinating actions. In the context of communications and computer engineering, a network typically consists of nodes (the communicating entities) and links (the physical or wireless media that connect them), governed by a set of rules called protocols that define how information is formatted, transmitted, and interpreted.

1.2.2 Objectives of Networks

Networks are designed to achieve a number of key objectives, including:

- Resource sharing: allowing multiple users or devices to share hardware, software, and data resources, reducing duplication and cost.
- Communication: enabling reliable and timely exchange of information between geographically distributed entities.
- Data centralization and access: allowing data generated at multiple points to be collected, stored, and accessed from a central or distributed location.
- Reliability and redundancy: ensuring that if one path or device fails, alternative paths or devices can maintain system operation.
- Scalability: supporting the addition of new devices or users without redesigning the entire system.

1.2.3 Advantages of Networks

The widespread adoption of networking technology is driven by several practical advantages:

- Improved efficiency through automated and remote data collection, reducing the need for manual intervention.
- Cost reduction by sharing infrastructure, hardware, and bandwidth among multiple users or devices.
- Faster decision-making, since real-time data exchange allows quicker responses to changing conditions.
- Flexibility and mobility, particularly in wireless networks, where devices are not constrained by physical cabling.
- Enhanced monitoring and control capabilities, which are particularly valuable in applications such as environmental monitoring, industrial automation, and security systems.

1.3 Types of Networks

Networks can be classified in multiple ways depending on scale, medium, and purpose. The following subsections describe the categories most relevant to this project, narrowing progressively from general computer networks down to the specific type of network used in this work. Figure 1.1 summarizes this classification path graphically.

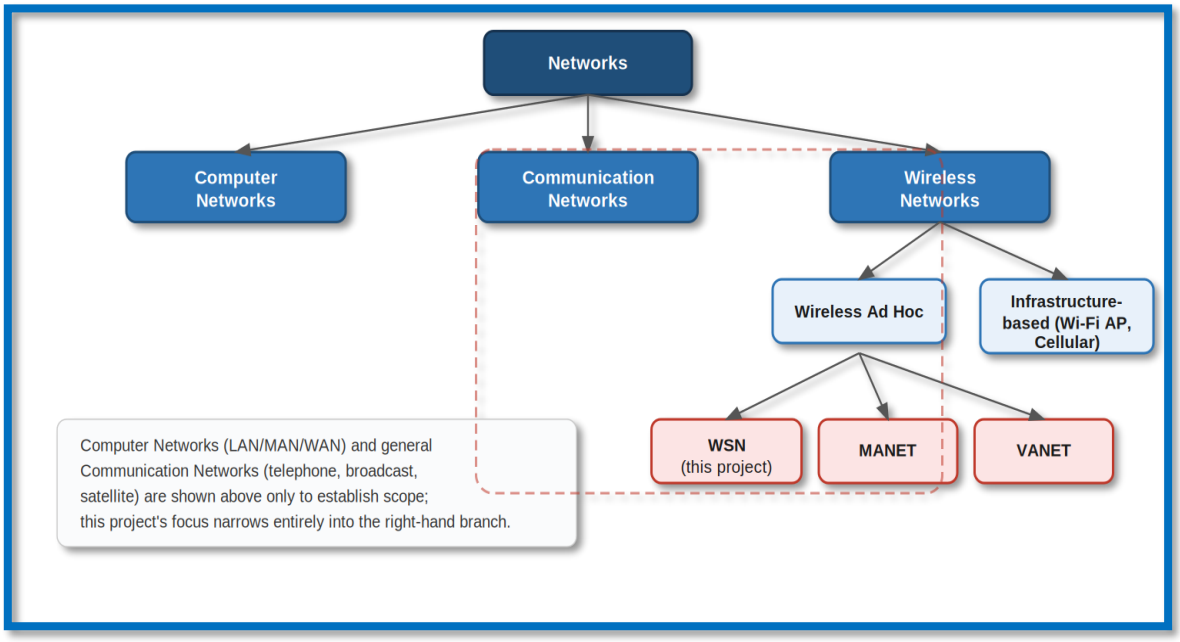

Networks
Computer Networks
Communication Networks
Wireless Networks
Wireless Ad Hoc
Infrastructure-based (Wi-Fi AP, Cellular)
WSN (this project)
MANET
VANET
Computer Networks (LAN/MAN/WAN) and general Communication Networks (telephone, broadcast, satellite) are shown above only to establish scope; this project's focus narrows entirely into the right-hand branch.

Figure 1.1: Classification of networks

1.3.1 Computer Networks

A computer network connects multiple computing devices to allow the sharing of data, files, and resources such as printers and storage. Computer networks are commonly classified by geographical scope into Local Area Networks (LAN), Metropolitan Area Networks (MAN), and Wide Area Networks (WAN). These networks typically rely on well-established infrastructure such as Ethernet cabling, routers, and switches, and are optimized for high throughput and general-purpose data exchange.

1.3.2 Communication Networks

Communication networks refer more broadly to any infrastructure designed to transmit information between two or more points, encompassing telephone networks, broadcast networks, satellite networks, and data networks. Unlike computer networks, which are primarily concerned with data exchange between computing devices, communication networks may carry voice, video, control signals, or sensor data, and can operate over wired or wireless media.

1.3.3 Wireless Networks

A wireless network is a communication network in which connections between nodes are established without physical cables, using electromagnetic waves (typically radio frequency) as the transmission medium. Wireless networks offer key advantages over wired networks, including ease of deployment, mobility support, and reduced installation cost, particularly in environments where cabling is impractical, such as forests, agricultural fields, or disaster zones. These advantages come at the cost of additional challenges, including signal attenuation, interference, and limited bandwidth compared to wired alternatives. As shown in Figure 1.1, wireless networks themselves split further into infrastructure-based networks (relying on fixed access points or cellular base stations) and wireless ad hoc networks, the branch directly relevant to this project and examined in detail in Sections 1.6 to 1.8.

1.3.4 Wireless Sensor Networks

A Wireless Sensor Network (WSN) is a specialized type of wireless network composed of a large number of small, low-cost, low-power sensor nodes that are spatially distributed to monitor physical or environmental conditions, such as temperature, humidity, pressure, or gas concentration, and to cooperatively relay this data through the network to a central collection point. WSNs are distinguished from general-purpose wireless networks by their emphasis on energy efficiency, autonomous operation, and data-centric communication rather than high-throughput general data transfer.

This project adopts the WSN paradigm directly: temperature, humidity, carbon monoxide (CO), and carbon dioxide (CO2) sensors are deployed as sensing nodes, with data acquired through National Instruments (NI) DAQ hardware and an NI WiFi communication module. This hardware effectively realizes the sensing and transceiver functions of a WSN node, while the acquired data is processed in LabVIEW and further analyzed by a Python-based decision module, illustrating how the general WSN concept introduced here is mapped onto a concrete forest fire detection application, as detailed in the chapters that follow.

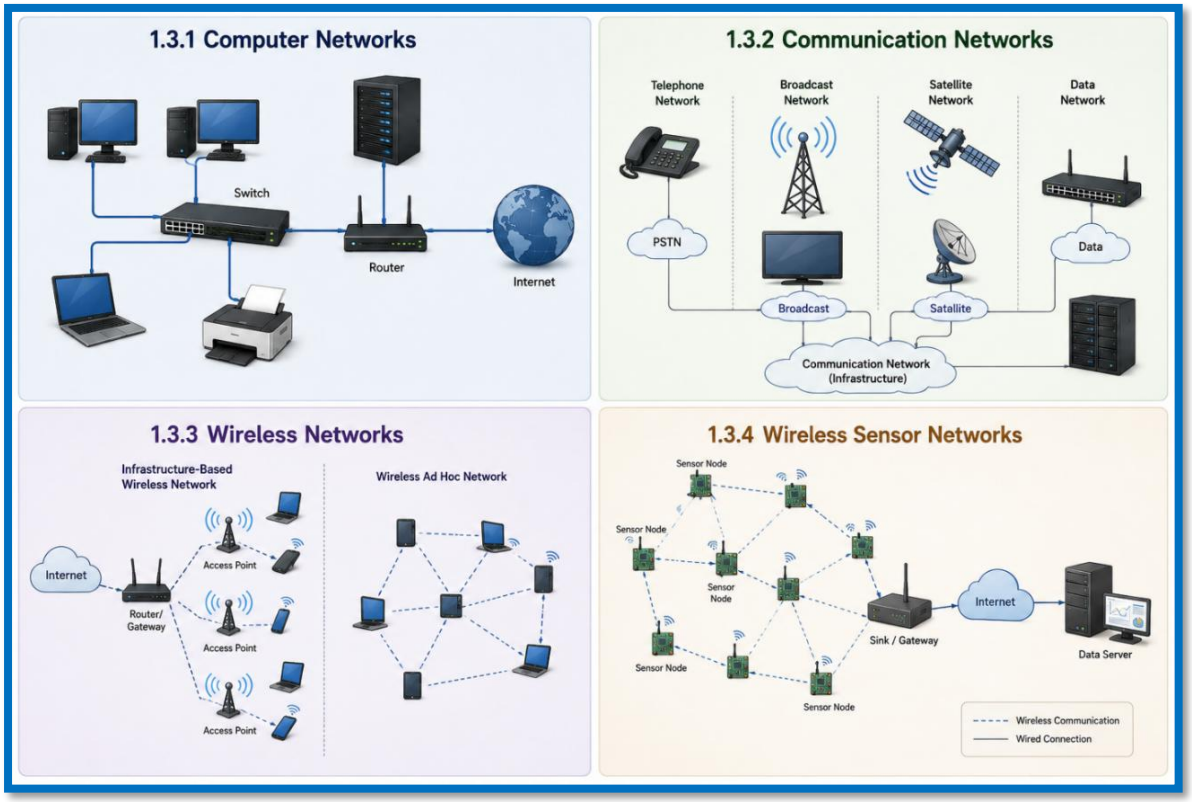


*Figure 1.2: **Types of networks***

1.4 Communication Systems

Regardless of whether a network is wired or wireless, every communication process can be described in terms of a generic communication system model consisting of three core components, illustrated in Figure 1.3.

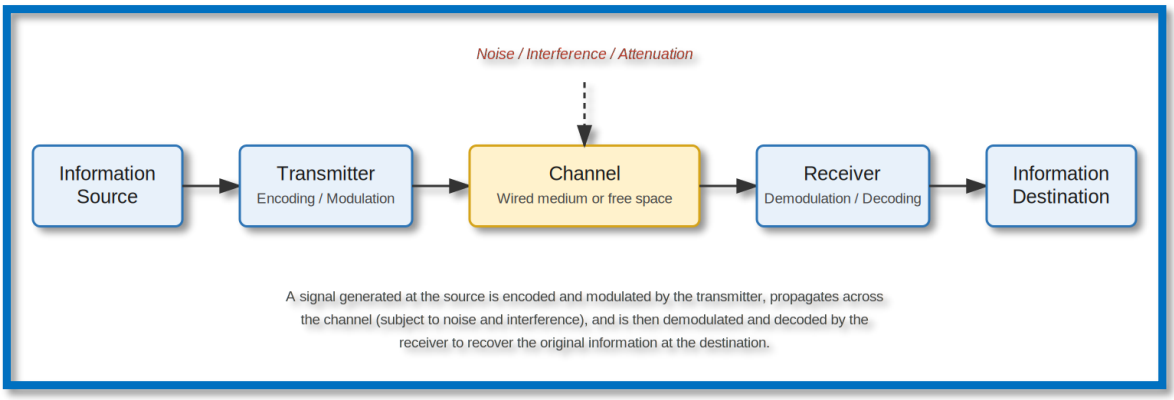

Noise / Interference / Attenuation
Information Source
Transmitter
Encoding / Modulation
Channel
Wired medium or free space
Receiver
Demodulation / Decoding
Information Destination
A signal generated at the source is encoded and modulated by the transmitter, propagates across the channel (subject to noise and interference), and is then demodulated and decoded by the receiver to recover the original information at the destination.

*Figure 1.3: **Generic communication system block diagram***

1.4.1 Communication System Components

Transmitter

The transmitter is responsible for converting the information signal (e.g., a sensor reading) into a form suitable for transmission over the communication channel. This typically involves encoding, modulation, and amplification of the signal so that it can propagate effectively through the chosen medium.

Channel

The channel is the physical medium through which the signal travels from the transmitter to the receiver. In wired systems, this may be a copper cable or optical fiber; in wireless systems, it is free space, through which electromagnetic waves propagate. The channel introduces impairments such as attenuation, noise, and interference, which must be accounted for in system design.

Receiver

The receiver captures the transmitted signal from the channel and performs the inverse operations of the transmitter, namely demodulation and decoding, to recover the original information signal, ideally with minimal distortion or error.

1.4.2 Wired versus Wireless Communication

The distinction between wired and wireless communication is one of the most fundamental in networking, and each approach carries its own profile of advantages, limitations, and best-fit use cases, illustrated conceptually in Figure 1.4.

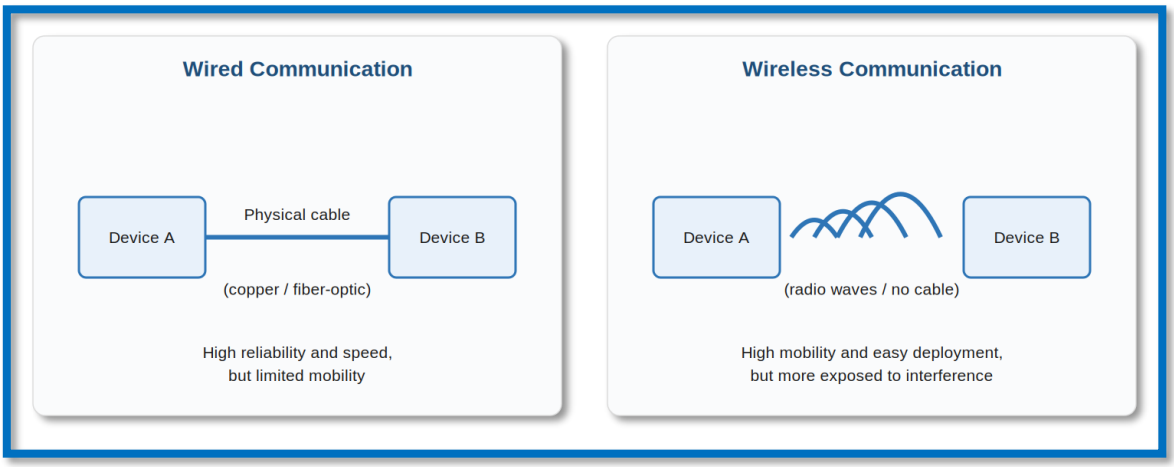


*Figure 1.4: **Wired versus wireless communication***

Wired communication carries data over a tangible physical medium, such as copper cable, coaxial cable, or optical fiber, with the two communicating devices joined by a direct physical link. Because the signal is confined to a guided medium, wired links typically offer high reliability, very high achievable data rates (fiber-optic links can reach hundreds of Gbps), strong inherent security (physical access is required to intercept the signal), and low, predictable latency. These benefits come at the cost of restricted mobility, since devices must stay physically connected; higher installation cost and effort for laying cable infrastructure; and limited scalability, since extending the network means running additional cabling.

Wireless communication instead carries data through the air as radio waves, microwaves, infrared, or satellite signals, using standardized protocols such as IEEE 802.11 (Wi-Fi), Bluetooth, Zigbee, and LTE/5G. This frees devices to move while remaining connected, allows networks to be deployed quickly without cabling, and scales easily since new devices can join without infrastructure changes, making wireless communication particularly suited to locations where running cables would be impractical, such as a forest environment. The trade-offs are a greater susceptibility to interference from other devices, obstacles, and weather; increased security risk, since signals can be intercepted over the air and must be protected with strong encryption (WPA2/WPA3); bandwidth that varies with distance and the number of connected users; and generally higher, less consistent latency than a wired link. Table 1.1 summarizes these differences.

Table 1.1: ***Comparison of wired and wireless communication***

Feature	***Wired***	***Wireless***
Medium	*Copper, fiber-optic cable*	*Radio waves, microwaves*
Speed	*Very high (up to ~100 Gbps)*	*Moderate (up to ~10 Gbps on Wi-Fi 6E)*
Mobility	*Limited — fixed devices*	*High — mobile devices*
Security	*High — physical access needed to intercept*	*Lower — depends on encryption strength*
Installation	*Complex cabling required*	*Easy, no cabling needed*
Interference	*Minimal*	*High (walls, devices, weather)*
Cost	*Higher infrastructure cost*	*Lower deployment cost*
Examples	*Ethernet, fiber, DSL*	***Wi-Fi, Bluetooth, 5G, Zigbee***

The system developed in this project relies on wireless communication, specifically WiFi-based transmission through the NI communication module, chosen for its ease of integration with the NI DAQ hardware and its suitability for short- to medium-range environmental monitoring, where the mobility and deployment-cost advantages of wireless transmission clearly outweigh the raw throughput advantage of a wired alternative.

1.5 Network Topologies

Network topology refers to the arrangement or structural layout of nodes and the connections between them. The choice of topology directly affects a network's cost, reliability, scalability, and fault tolerance. The most common topologies are described below and illustrated in Figure 1.5.

- Bus topology: all nodes are connected to a single shared communication line (the bus). It is simple and low-cost but suffers from a single point of failure: if the bus is damaged, the entire network fails.
- Ring topology: each node is connected to exactly two neighboring nodes, forming a closed loop. Data travels around the ring until it reaches its destination. While efficient for predictable data flow, a single broken link can disrupt the entire ring unless redundant paths are provided.
- Star topology: all nodes are connected individually to a central hub or controller. This topology is easy to manage and isolate faults in, since a failure in one node does not affect others, but the central hub represents a critical point of failure.
- Mesh topology: nodes are interconnected with multiple redundant paths between them, offering high reliability and fault tolerance at the cost of increased wiring/connection complexity and cost.

- Tree topology: nodes are arranged hierarchically, with groups of star-configured nodes connected to a central backbone, combining some of the manageability of star topology with the scalability needed for larger networks.

Wireless Sensor Networks most commonly employ star or mesh topologies, or hybrid combinations such as the cluster-tree topology shown in Figure 1.5. In a star-based WSN, all sensor nodes communicate directly with a central sink or base station, which suits small-scale or single-hop deployments and matches the configuration used in this project, where each sensor node interfaces directly with the NI DAQ/WiFi module acting as the central collection point. In larger-scale WSN deployments, a mesh or cluster-tree topology is often preferred, since it allows sensor nodes to relay data through neighboring nodes over multiple hops, extending coverage and improving fault tolerance when nodes are deployed across a wide forest area beyond the direct range of a single base station.

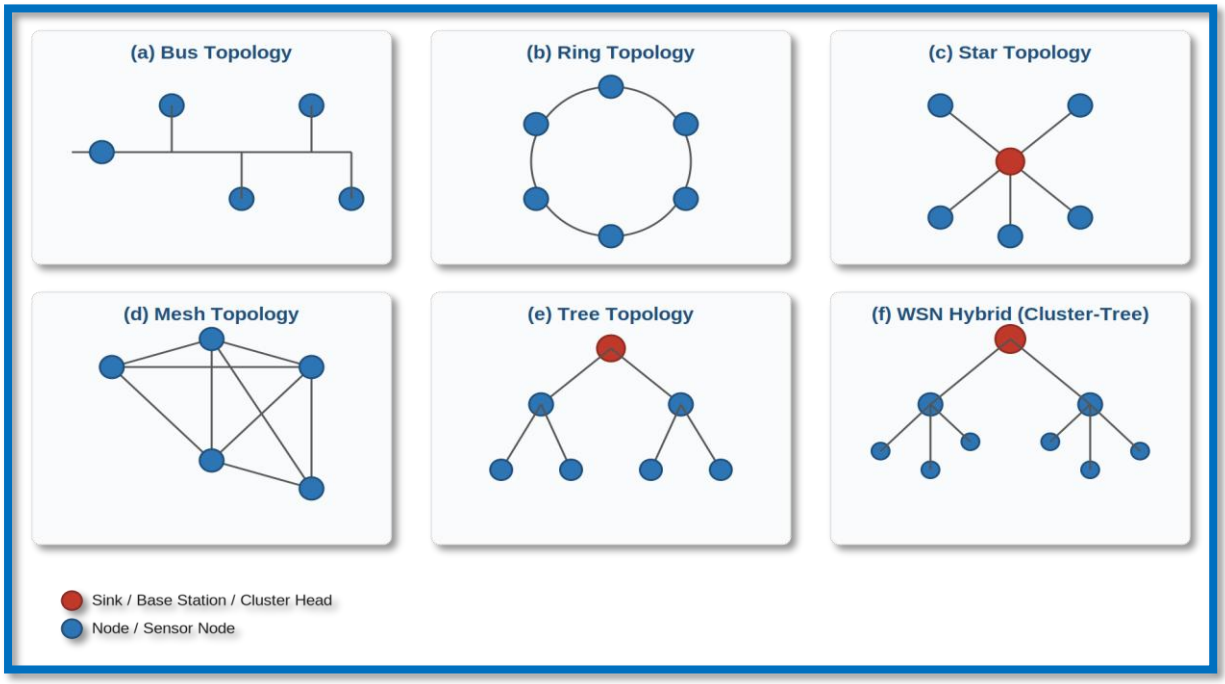


Figure 1.5: ***Common network topologies***

1.6 Wireless Sensor Networks

1.6.1 Definition

As introduced in Section 1.3.4, a Wireless Sensor Network is a distributed network of small, low-power devices called sensor nodes (or motes), capable of sensing physical or environmental conditions, such as temperature, humidity, pressure, motion, or light, and transmitting the collected data wirelessly to a central point known as a base station or sink node. WSNs emerged as a natural evolution of wireless communication and are today considered a cornerstone of the Internet of Things (IoT), enabling the monitoring and control of physical environments without requiring fixed infrastructure.

1.6.2 Architecture

A WSN architecture is generally organized into three conceptual layers. The physical layer governs the actual radio transmission between nodes and the base station, using technologies such as IEEE 802.15.4, ZigBee, WiFi, or Bluetooth depending on application requirements. The data link layer is responsible for establishing reliable links between neighboring nodes and managing medium access to avoid collisions. Above these, the network and application layers handle routing of data toward the sink and the interpretation/use of that data by the end application, respectively.

Beyond this layered communication view, it is useful to consider the system-level architecture of a WSN deployment as a whole, shown in Figure 1.6. A set of sensor nodes is distributed across the monitored field; rather than each node necessarily reaching the sink directly, data is typically relayed hop-by-hop between neighboring nodes until it reaches the sink (or aggregator) node. The sink then forwards the aggregated data, via a gateway or wireless link, to a back-end application layer. In this project, that back-end layer is realized concretely by LabVIEW (acquisition and processing), the Python decision module (fire-detection logic), and Firebase (real-time alerting to the end user), mapping the general WSN system architecture directly onto the system implemented in this report.

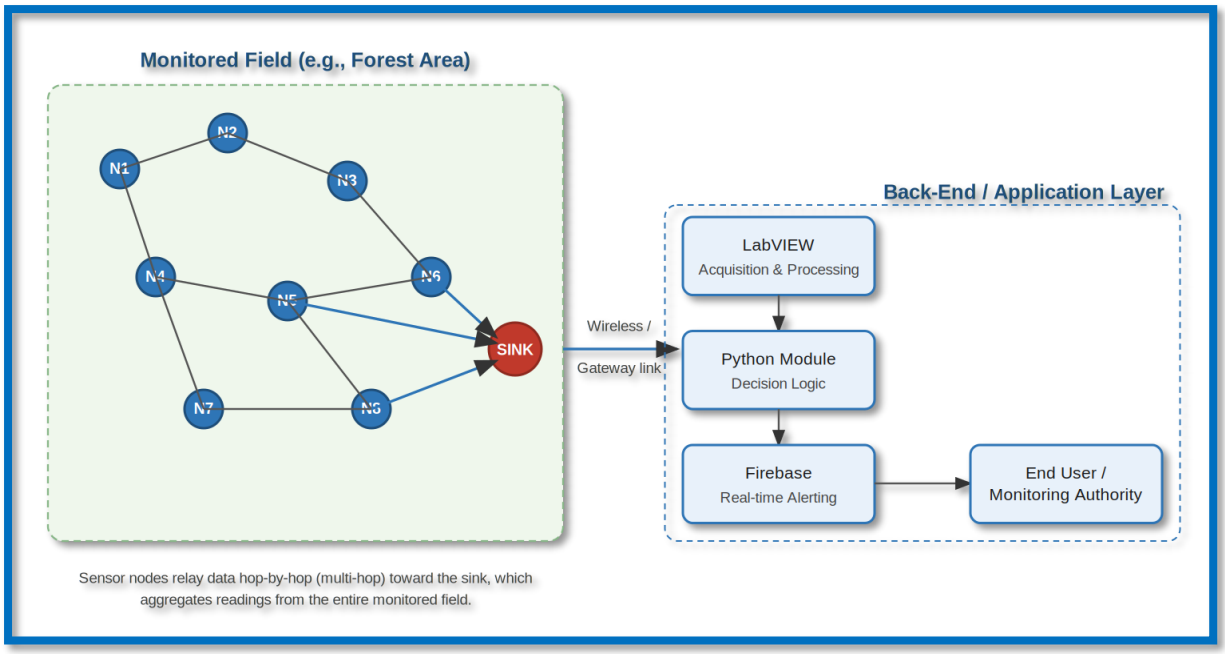


Figure 1.6: System-level architecture of a WSN deployment

1.6.3 Components

Each individual sensor node, regardless of the overall network architecture, is composed of four principal functional units, as illustrated in Figure 1.7:

- Sensing unit: comprises the physical sensor and an analog-to-digital converter (ADC), responsible for capturing a physical quantity (e.g., temperature, humidity, gas concentration) and converting it into a digital signal.
- Processing unit: typically a microcontroller or microprocessor with associated memory, responsible for local data processing, basic decision-making, and control of the other units.
- Transceiver (communication) unit: a radio transceiver responsible for wireless transmission and reception of data to and from other nodes or the sink.
- Power unit: supplies energy to all other units, typically a battery, sometimes supplemented by energy harvesting (e.g., solar cells) in field deployments.
- Optional units: depending on the application, a sensor node may also include a location-finding system (e.g., GPS) or a mobilizer for repositioning, though these are not required in static monitoring deployments such as this project.

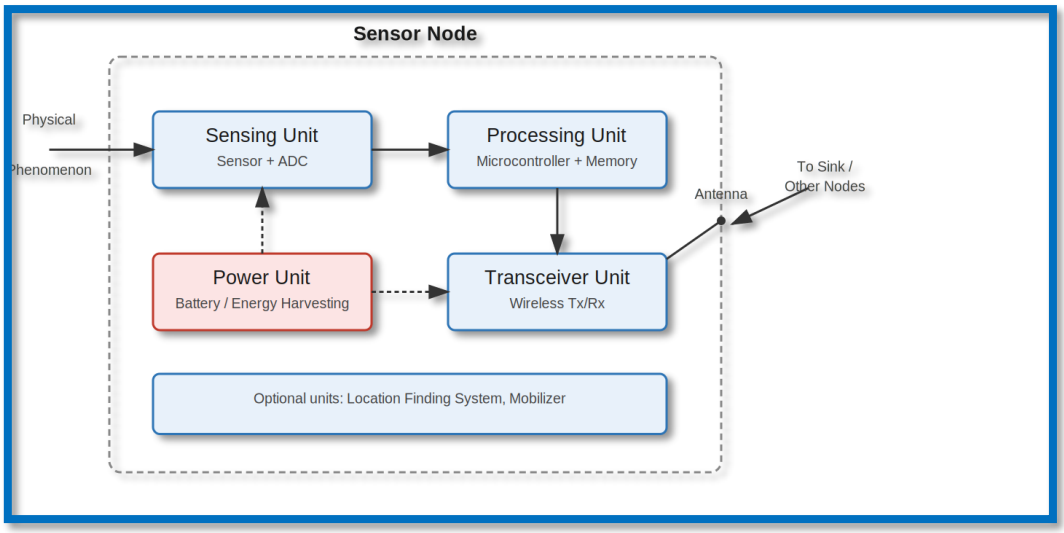


Figure 1.7: ***Functional block diagram of a wireless sensor node***

1.6.4 Characteristics

WSNs exhibit a distinctive set of characteristics that differentiate them from general-purpose wireless networks:

- Large-scale deployment: a WSN may consist of hundreds or even thousands of sensor nodes deployed over a wide area, providing high spatial sampling resolution.
- Energy constraints: sensor nodes are typically battery-powered, making energy efficiency the most critical design consideration at every layer of the network.
- Self-organization: nodes are capable of automatically forming a network upon deployment without requiring manual configuration.
- Multi-hop communication: data is relayed from node to node until it reaches the sink, extending the effective range of the network.
- Data-centric operation: communication and routing in WSNs are often organized around the data itself rather than fixed node addresses, since the data is generally of greater interest than the identity of the reporting node.
- Limited resources: individual nodes have constrained processing power, memory, and bandwidth compared to conventional computing or networking devices.
- Fault tolerance: the network continues to function even if individual nodes fail, owing to redundancy in node deployment.

1.7 Ad Hoc Networks

1.7.1 Definition

An ad hoc network is a decentralized, infrastructure-less wireless network in which nodes communicate directly with one another and collaboratively perform routing functions, without relying on a fixed access point, base station, or pre-existing network infrastructure. Each node in an ad hoc network can act simultaneously as both an end device and a router, forwarding data on behalf of other nodes to extend network coverage beyond direct radio range. The term itself derives from the Latin ad hoc, meaning "for this purpose," reflecting the on-demand, temporary character of these networks. Ad hoc networking underlies many modern wireless technologies, including Bluetooth mesh, Zigbee mesh, and mobile peer-to-peer networks, and is closely related to WSNs, as explored in Section 1.8.

Figure 1.8 illustrates the defining advantage of ad hoc operation: when a destination node lies beyond the direct radio range of a source node, a conventional point-to-point link simply fails, whereas an ad hoc network can still deliver the data by relaying it through one or more intermediate nodes, each acting as both host and router.

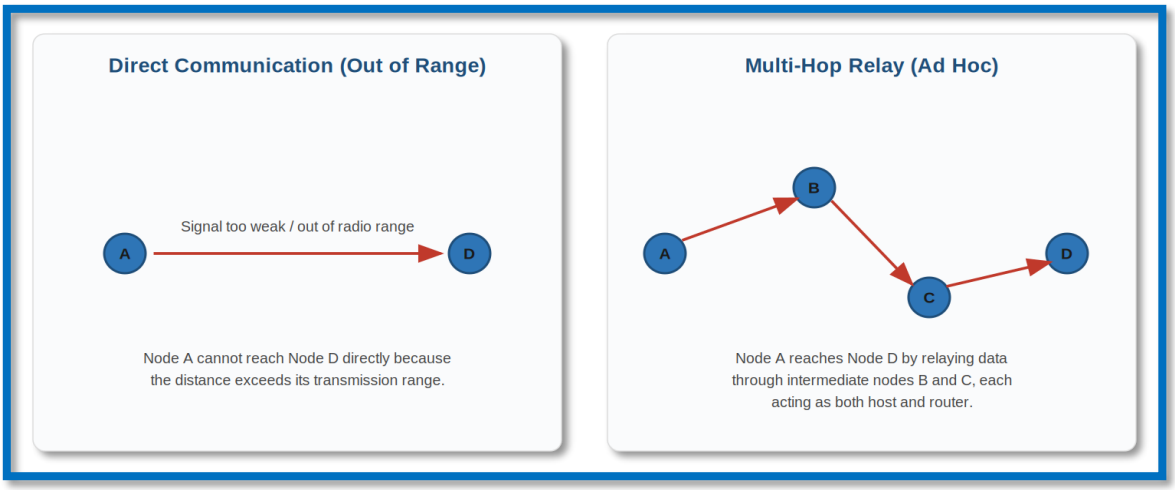

Direct Communication (Out of Range)
Signal too weak / out of radio range
A
D
Node A cannot reach Node D directly because the distance exceeds its transmission range.
Multi-Hop Relay (Ad Hoc)
A
B
C
D
Node A reaches Node D by relaying data through intermediate nodes B and C, each acting as both host and router.

Figure 1.8: ***Direct communication failing beyond radio range***

1.7.2 Features

- Infrastructure-less operation: the network can be established anywhere, at any time, without dependence on pre-installed access points or base stations; all nodes act as equal peers.
- Self-configuring topology: nodes automatically discover neighbors and configure routing without manual intervention.
- Dynamic topology: since nodes may move, join, or leave the network, the network topology can change frequently during operation.

- Distributed operation: there is no central controller; every node participates in routing and forwarding decisions.
- Multi-hop routing: data may pass through several intermediate nodes before reaching its destination, as shown in Figure 1.8.
- Resource limitations: nodes typically have constrained battery life, processing power, and bandwidth.

1.7.3 Types of Ad Hoc Networks

Two major categories of ad hoc networks are widely recognized in the literature. A Mobile Ad Hoc Network (MANET) consists of mobile devices, such as laptops or smartphones, that communicate directly with one another; MANETs are commonly used in disaster recovery, military operations, and temporary events where fixed infrastructure is unavailable. A Vehicular Ad Hoc Network (VANET) is a specialized form of MANET designed for vehicles, where cars communicate with each other (vehicle-to-vehicle, V2V) and with roadside infrastructure (vehicle-to-infrastructure, V2I) to support applications such as collision avoidance, traffic management, and autonomous-driving assistance. As will be discussed in Section 1.8, WSNs constitute a third major category within the same ad hoc family, distinguished by static deployment and a sensing-driven purpose rather than user mobility.

1.7.4 Advantages

- Rapid and low-cost deployment, since no fixed infrastructure needs to be installed in advance.
- High flexibility, allowing the network to be deployed in remote, temporary, or hard-to-reach locations such as forests or disaster sites.
- Improved coverage through multi-hop relaying, extending the effective range of the network beyond a single node's transmission radius.
- Resilience to the failure of any single node, owing to the decentralized and self-healing nature of routing.

1.7.5 Challenges

- Limited bandwidth and range, since nodes typically rely on low-power radios suitable for short- to medium-range communication.
- Energy constraints, particularly for battery-powered nodes that must also expend energy forwarding other nodes' traffic.

- Security vulnerabilities, since the decentralized, infrastructure-less nature of ad hoc networks makes them more susceptible to eavesdropping, spoofing, and routing attacks.
- Routing complexity, as dynamic topologies require adaptive routing protocols capable of responding quickly to network changes.
- Scalability limitations, since performance can degrade significantly as the number of nodes or the physical distance between them increases.

The differences between conventional infrastructure-based networks and ad hoc networks are summarized in Table 1.2.

***Table 1.2:* Infrastructure-based networks versus ad hoc networks**

Feature	***Infrastructure-Based***	***Ad Hoc***
Infrastructure	*Requires fixed access point / router*	*No fixed infrastructure*
Setup	*Pre-configured*	*Self-configuring*
Topology	*Fixed (typically star)*	*Dynamic, changes over time*
Scalability	*High, via additional access points*	*Limited as node count/distance grows*
Range	*Extended via access points*	*Limited by node proximity / multi-hop*
Typical use cases	*Offices, homes, campuses*	*Military, disaster response, WSN*

Ad hoc networks play a critical role in scenarios where deploying traditional infrastructure is impractical, such as disaster relief operations, battlefield communications, and remote sensor deployments, and, as Table 1.2 indicates, they form the underlying architecture for WSNs, since sensor nodes typically communicate in an ad hoc fashion to relay data back to the base station.

1.8 Relationship Between WSN and Ad Hoc Networks

The relationship between Wireless Sensor Networks and Ad Hoc networks is central to the title and conceptual framing of this graduation project, “Application on Ad Hoc Network Using WSN,” and therefore merits particular attention. This section first revisits how the two concepts relate at the theoretical level, then examines, with full transparency, how closely the system implemented in this project actually follows ad hoc principles in practice.

1.8.1 WSN as a Subclass of Ad Hoc Networks

Wireless Sensor Networks are widely regarded in the literature as a specialized subclass of wireless ad hoc networks, alongside other subclasses such as MANET and VANET, as illustrated in Figure 1.9. Both share fundamental characteristics: radio-based communication, decentralized and self-organized operation, and the absence of a pre-existing wired infrastructure connecting the nodes. In both network types, nodes are deployed without requiring a fixed communication backbone, and the network is expected to organize itself, at least in principle, without manual configuration.

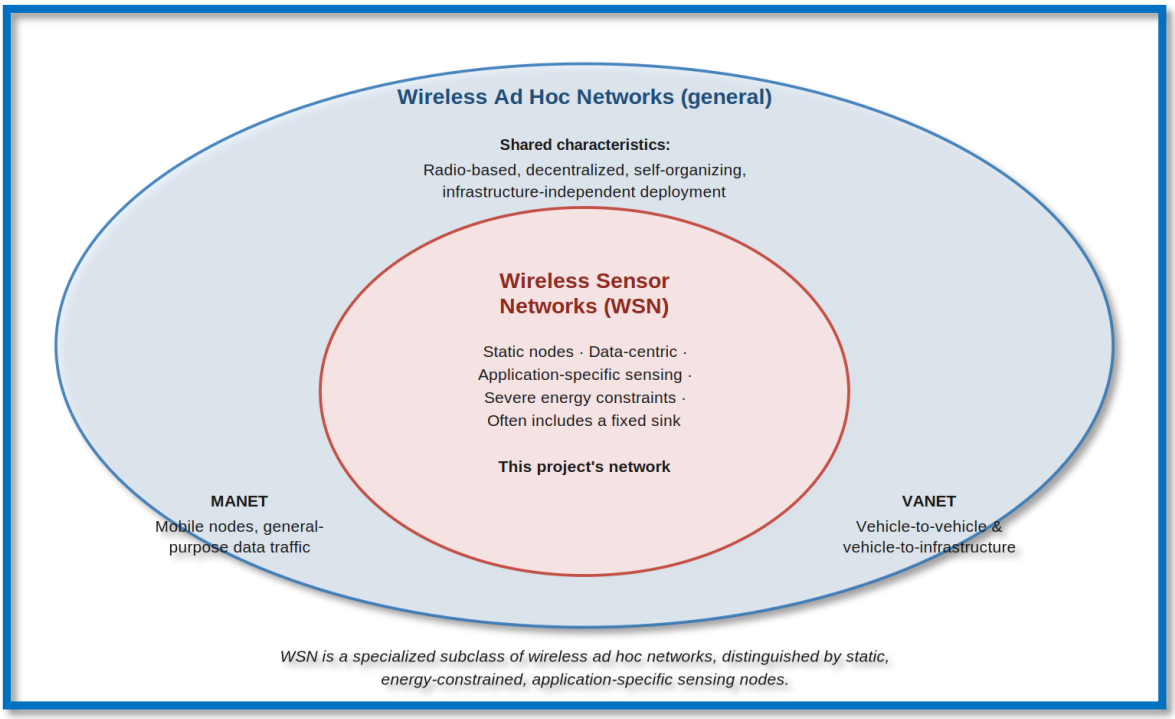


Figure 1.9: WSN as a specialized subclass of wireless ad hoc networks

1.8.2 Point-by-Point Comparison

Table 1.3 compares WSN and (classical) Ad Hoc networks across the dimensions most relevant to this project, distilling the similarities and differences discussed throughout this chapter into a single reference.

Table 1.3: ***WSN versus classical Ad Hoc networks***

Aspect	***Wireless Sensor Network (WSN)***	***Classical Ad Hoc Network (e.g., MANET)***
Node mobility	*Typically static once deployed*	*Nodes are mobile by design*
Purpose	*Data-centric, built around one sensing task*	*General-purpose data communication between users/devices*
Energy profile	*Severe, long-term battery constraints*	*Constrained, but typically less extreme and shorter-lived*
Presence of a sink	*Frequently includes a fixed sink/base station*	*Often fully distributed, with no fixed node*
Routing	*May be single-hop (star) or multi-hop, depending on deployment scale*	*Defining feature: multi-hop, peer-relayed routing*
Infrastructure dependency	*Infrastructure-independent at deployment, though a sink is fixed*	*Fully infrastructure-independent, no fixed node at all*
Self-organization	*Present in large, dense deployments; optional in small ones*	*Core requirement, since routes must be discovered dynamically*

Table 1.3 makes clear that WSN and Ad Hoc networks overlap strongly in philosophy, infrastructure-independence, decentralized deployment, self-organization, but diverge specifically on mobility, routing complexity, and the role of a fixed sink. This divergence is precisely what needs to be examined honestly when relating the general theory to the system actually built in this project.

1.8.3 How This Project's Network Relates to Ad Hoc Principles

It is important to state plainly how the physical system implemented in this project compares against the theoretical picture above, rather than only asserting the relationship at a conceptual level. As described in Chapter 3, every sensor node (temperature, humidity, CO, and CO2) communicates wirelessly and directly with the central NI DAQ/WiFi module, with no node ever relaying another node's data on its way to the module. In topology terms, this is a pure star, single-hop arrangement, not a multi-hop relay network. Figure 1.10 contrasts this actual configuration against a genuine ad hoc network in which some nodes are out of the sink's direct range and must rely on neighboring nodes to forward their data.

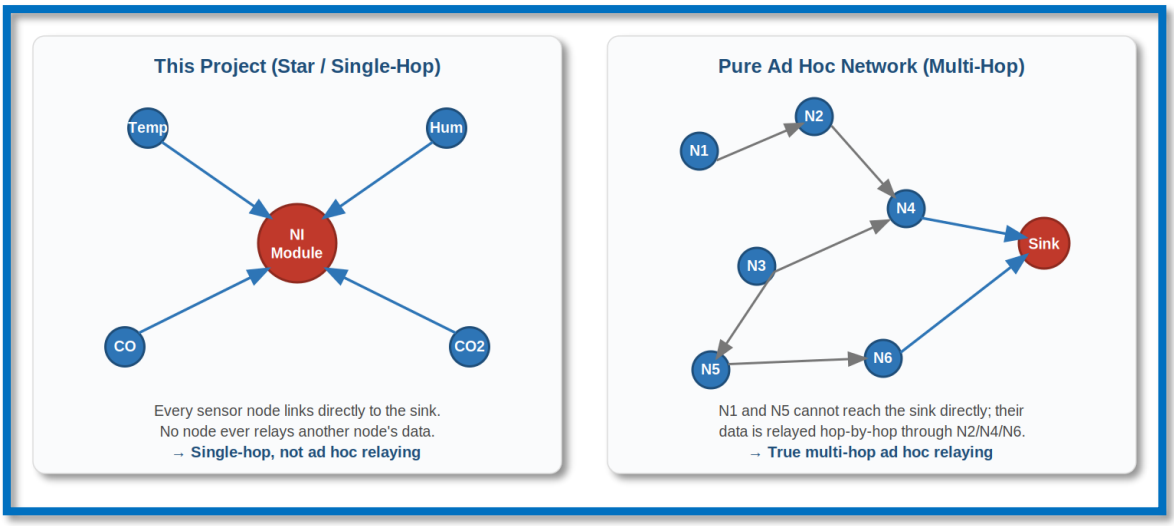


Figure 1.10: ***actual star/single-hop topology versus pure ad hoc multi-hop topology***

This distinction matters because multi-hop relaying, illustrated earlier in Figure 1.8, is often treated as the single most defining behavioral feature of an ad hoc network: it is what allows the network to extend coverage beyond any one node's radio range without fixed infrastructure. Strictly by that narrow definition, the implemented system does not exhibit ad hoc routing, since the small number of sensors and the limited deployment area mean that every node already falls within direct range of the NI module, leaving no need for relaying.

Nevertheless, the system legitimately satisfies the broader set of ad hoc characteristics that do not depend specifically on multi-hop routing. It requires no pre-existing wired backbone connecting the sensors. It can be set up and torn down in a given location without installing fixed access points. Each node operates independently and transmits opportunistically rather than on a centrally scheduled wired bus. These are exactly the infrastructure-independence and decentralized-deployment properties listed in Table 1.3 as shared between WSN and Ad Hoc networks, and they hold for this system regardless of whether multi-hop relaying is present.

The most accurate and defensible framing, therefore, is the following: the project is a Wireless Sensor Network deployed in an ad hoc manner, in the sense of being infrastructure-independent and self-contained, but it operates at single-hop scale rather than exercising the multi-hop relaying that characterizes ad hoc networks in their fullest sense. This is not a flaw in the implementation; it is a direct and reasonable consequence of the deployment scale chosen for this graduation project, where a small number of sensors monitoring a limited area has no practical need for relaying. The architecture is, however, deliberately extensible: because each sensor node already behaves as an independent wireless transmitter rather than a wired endpoint, scaling the deployment to a wider forest area in future work (see Chapter 7)

would only require adding genuine multi-hop routing logic between sensor nodes, without altering the fundamental sensing and decision-making pipeline described in this report. In that sense, the system is best described as ad hoc-ready rather than ad hoc-complete, fully consistent with the project title while remaining precise about what was actually implemented and tested.

1.9 Applications of WSN

The flexibility, low cost, and self-organizing nature of WSNs have enabled their adoption across a wide range of domains:

Agriculture

WSNs are used in precision agriculture to monitor soil moisture, temperature, and nutrient levels across large fields, enabling optimized irrigation, fertilization, and crop management with minimal manual intervention.

Healthcare

In healthcare, wearable and implantable sensor nodes form body area networks that continuously monitor physiological parameters such as heart rate, blood oxygen levels, and temperature, supporting remote patient monitoring and early detection of medical emergencies.

Smart Cities

Smart city deployments use WSNs for applications such as traffic monitoring, smart lighting, waste management, and air quality assessment, contributing to more efficient and responsive urban infrastructure.

Industrial Automation

In industrial settings, WSNs support machine condition monitoring, predictive maintenance, and general factory automation, reducing downtime and enabling data-driven maintenance scheduling without the cost of wiring an entire production floor.

Environmental Monitoring

Environmental monitoring is one of the most established WSN application domains, encompassing the tracking of temperature, humidity, air quality, water quality, and seismic activity across wide geographic areas that would be difficult or costly to monitor using wired infrastructure.

Forest Fire Detection

Forest fire detection represents a particularly demanding and impactful environmental monitoring application, and is the specific application addressed by this graduation project.

By distributing temperature, humidity, CO, and CO2 sensor nodes across a forested area and continuously relaying their readings to a central decision-making system, a WSN-based fire detection system can identify the characteristic environmental signature of a developing fire, rising temperature, falling humidity, and increasing combustion gas concentrations, far earlier than traditional methods such as watchtower observation or satellite imagery. This capability directly motivates the system design developed in the remainder of this report, in which NI DAQ hardware, LabVIEW, Python, and Firebase are combined to realize a complete, real-time WSN-based fire detection and alerting pipeline.

1.10 Chapter Summary

This chapter established the theoretical foundation for the remainder of the report. It began with a general definition of networks, their objectives, and their advantages, before narrowing progressively through computer networks, communication networks, wireless networks, and finally Wireless Sensor Networks. The fundamental components of a communication system, transmitter, channel, and receiver, were introduced, along with a detailed comparison of wired and wireless communication. Common network topologies were reviewed, with particular attention to the star and cluster-tree hybrid topologies favored by WSN deployments. The architecture (including the system-level view of sensor nodes, sink, and back-end application), components, and characteristics of WSNs were then described in detail, followed by an examination of ad hoc networks, their MANET and VANET subtypes, and, critically, the relationship between WSNs and ad hoc networks that underpins the framing of this project. Finally, key application domains of WSN technology were surveyed, culminating in forest fire detection, the specific application context developed throughout the remainder of this report. Chapter 2 builds on this foundation by examining Wireless Sensor Networks in greater technical depth.

CHAPTER

2

Wireless Sensor Networks (WSN)

- Network Topologies
- Wireless Communication Protocols
- Routing & Data Aggregation
- Architecture & Concentrator
- Environmental & Metrological Monitoring
- Industrial & Real-World Applications

2.1 Introduction

Wireless Sensor Networks (WSNs) have emerged as an important technology for monitoring and collecting data from distributed environments. A WSN consists of multiple sensor nodes capable of sensing, processing, and transmitting information wirelessly to a central gateway. These networks enable real-time monitoring without the need for extensive wiring infrastructure, making them suitable for a wide range of industrial, environmental, and research applications.

Recent advances in wireless communication, embedded systems, and low-power electronics have significantly enhanced the capabilities of WSNs. Modern wireless sensor networks can operate for extended periods while providing reliable communication and continuous data acquisition. Their flexibility, scalability, and ease of deployment have led to widespread adoption in environmental monitoring, industrial automation, agriculture, and smart infrastructure systems.

The proposed project utilizes a Wireless Sensor Network to provide distributed environmental monitoring. The network is based on National Instruments (NI) WSN devices, which offer reliable wireless communication, low power consumption, and industrial-grade performance.

2.2 Wireless Sensor Networks (WSNs)

A Wireless Sensor Network is a collection of spatially distributed sensor nodes that cooperate to monitor physical or environmental conditions and transmit the acquired information to a central collection point. The sensor nodes communicate wirelessly, eliminating the need for traditional wired communication infrastructure.

Each node within the network typically performs three primary functions:

1. Data Acquisition
2. Data Processing
3. Wireless Communication

The acquired information is transmitted through the wireless network to a gateway, which serves as the interface between the sensor network and external monitoring systems.

Wireless Sensor Networks are particularly advantageous in applications where wired installations are expensive, impractical, or impossible. The wireless architecture reduces installation complexity while providing flexibility for future expansion.

The basic operation of a WSN can be summarized as follows:

Sensor nodes collect data from the surrounding environment.

The collected data are processed locally.

Information is transmitted wirelessly to the gateway.

The gateway receives and organizes the incoming data.

The data are forwarded to monitoring or analysis systems.

Through this process, WSNs enable continuous and real-time monitoring of distributed environments.

2.3 Components of Wireless Sensor Networks

A Wireless Sensor Network consists of several interconnected components that work together to acquire, transmit, and manage data.

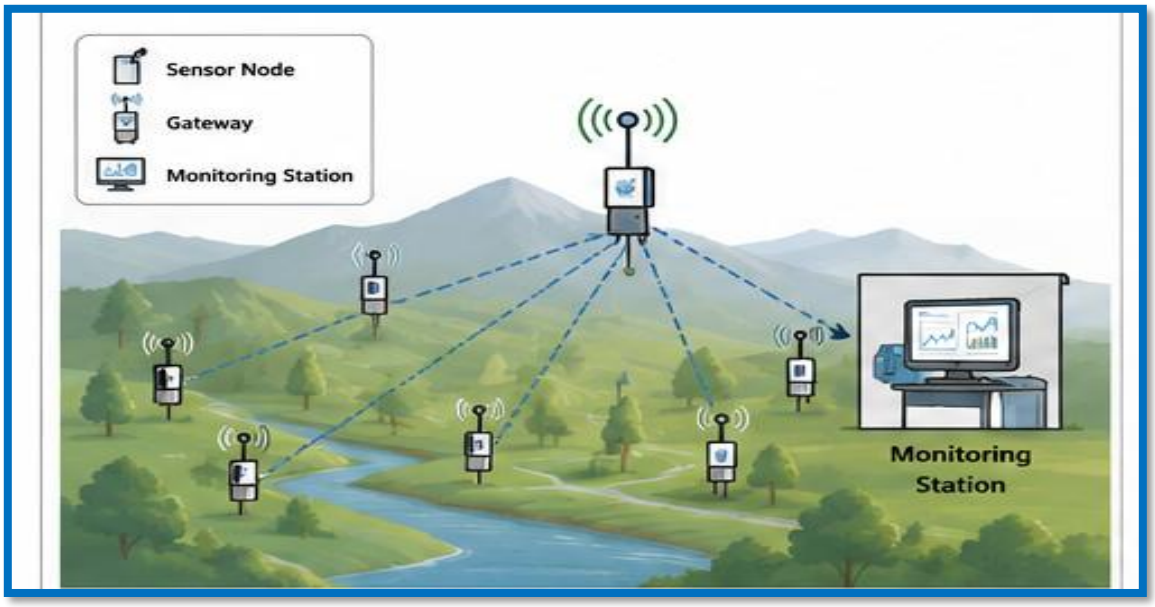


Figure 2.1: General Architecture of a Wireless Sensor Network

2.3.1 Sensor Nodes

Sensor nodes represent the primary data acquisition elements of the network. These devices collect information from the monitored environment and transmit the collected data through the wireless network.

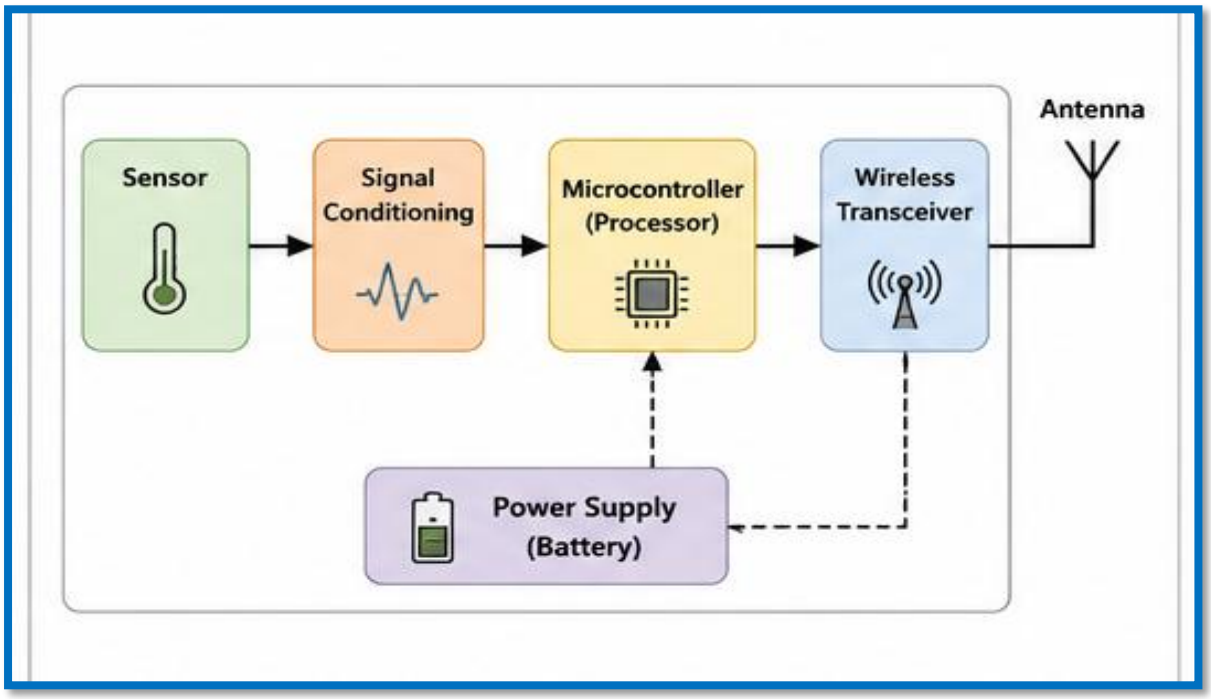


Figure 2.2: Basic Components of a Wireless Sensor Node

A typical sensor node contains:

Sensing Unit

Signal Conditioning Circuitry

Processing Unit

Memory

Wireless Transceiver

Power Supply

Sensor nodes are designed to operate with minimal power consumption while maintaining reliable communication capabilities.

2.3.2 Gateway

The gateway serves as the central communication hub of the network. It receives data from multiple wireless nodes and forwards the information to external monitoring systems.

The gateway is responsible for:

Network Coordination

Data Collection

Communication Management

Device Registration

Network Monitoring

As the central component of the network, the gateway ensures reliable communication between sensor nodes and monitoring applications.

2.3.3 Communication Network

The communication network provides the wireless link between sensor nodes and the gateway. Communication is typically performed using low-power wireless protocols designed for sensor network applications.

The communication network must provide:

Reliable Data Transmission

Low Power Consumption

Network Scalability

Communication Security

Interference Resistance

These characteristics ensure stable operation under various environmental conditions.

2.3.4 Monitoring Station

The monitoring station represents the end-user interface of the system. It receives information from the gateway and presents the collected data in a meaningful format.

Monitoring stations may perform:

Data Visualization

Data Storage

Data Analysis

Alarm Generation

Report Generation

This component enables users to monitor system status and evaluate environmental conditions.

2.4 Wireless Sensor Network Architecture

The architecture of a Wireless Sensor Network defines how information flows through the system. Although implementations may vary, most WSNs can be represented using a layered architecture.

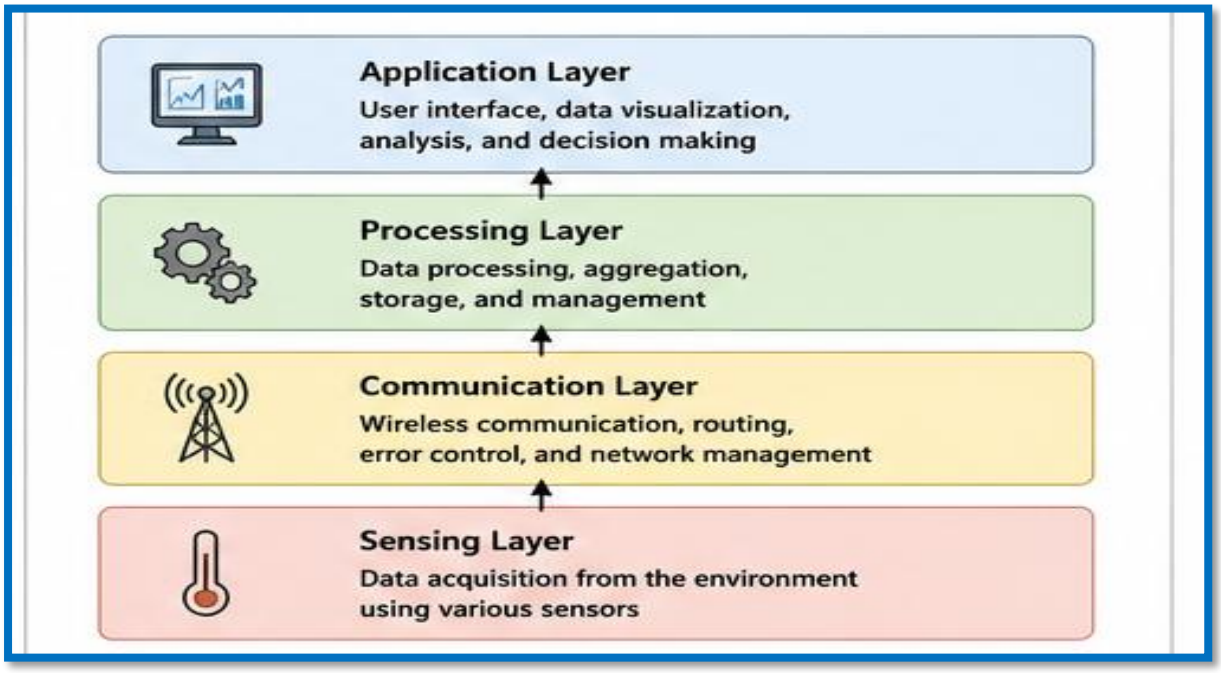

Application Layer
User interface, data visualization, analysis, and decision making
Processing Layer
Data processing, aggregation, storage, and management
Communication Layer
Wireless communication, routing, error control, and network management
Sensing Layer
Data acquisition from the environment using various sensors

Figure 2.3: Wireless Sensor Network Layered Architecture

2.4.1 Sensing Layer

The sensing layer is responsible for acquiring physical information from the environment. It represents the interface between the physical world and the electronic system.

The sensing layer performs:

Data Acquisition

Signal Detection

Initial Measurement Collection

The quality of the acquired information directly influences overall system performance.

2.4.2 Communication Layer

The communication layer manages wireless data transmission between nodes and the gateway.

Its primary responsibilities include:

Packet Transmission

Packet Reception

Error Detection

Network Access Control

Wireless Communication Management

Reliable operation of this layer is essential for successful data delivery.

2.4.3 Processing Layer

The processing layer is responsible for handling acquired information before transmission.

Typical functions include:

Data Processing

Data Formatting

Temporary Storage

System Control Functions

Local processing can reduce communication overhead and improve network efficiency.

2.4.4 Application Layer

The application layer provides interfaces for monitoring, visualization, and system management.

Functions performed at this layer include:

Data Presentation

User Interaction

Data Analysis

System Configuration

This layer allows users to access and interpret the information generated by the network.

2.5 Characteristics of Wireless Sensor Networks

Wireless Sensor Networks possess several characteristics that distinguish them from conventional monitoring systems.

2.5.1 Low Power Consumption

Power efficiency is a fundamental requirement for wireless sensor networks. Many sensor nodes operate using batteries and therefore must minimize energy consumption.

Low-power operation is achieved through:

Sleep Modes

Efficient Communication Protocols

Optimized Sampling Strategies

This enables long-term autonomous operation.

2.5.2 Wireless Connectivity

Wireless communication eliminates the need for extensive cabling infrastructure and simplifies system deployment.

Benefits include:

Reduced Installation Cost

Increased Deployment Flexibility

Simplified Maintenance

Remote Accessibility

Wireless connectivity is one of the primary advantages of WSN technology.

2.5.3 Scalability

Wireless sensor networks can be expanded by adding new nodes without major modifications to the existing infrastructure.

Scalability allows:

Network Growth

Increased Coverage Area

Flexible System Expansion

This capability is particularly useful in large-scale monitoring applications.

2.5.4 Reliability

Reliable operation is essential for monitoring systems that depend on continuous data collection.

Reliability is achieved through:

Error Detection Mechanisms

Robust Communication Protocols

Network Monitoring Functions

These features improve communication stability and data integrity.

2.5.5 Self-Organization

Many wireless sensor networks can automatically configure communication relationships between network devices.

Self-organization simplifies deployment and reduces the need for extensive manual configuration.

2.5.6 Flexibility

Wireless sensor networks can be adapted to various monitoring applications and operating environments.

This flexibility supports deployment in:

Environmental Monitoring

Agriculture

Industrial Automation

Infrastructure Monitoring

Research Applications

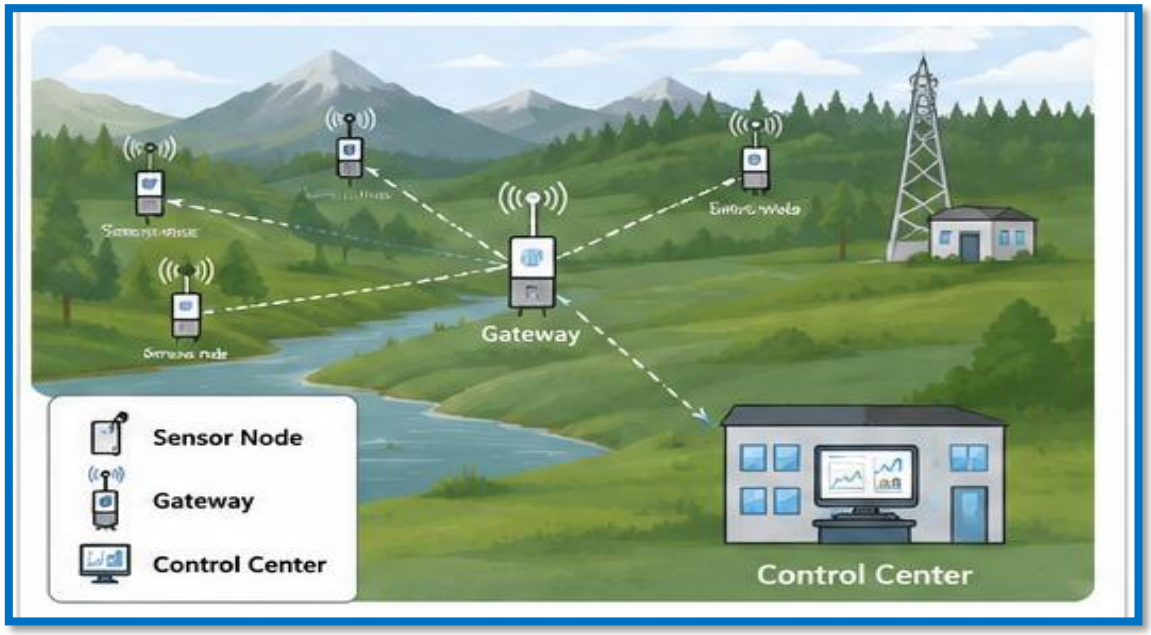


Figure 2.4: Environmental Monitoring Using Wireless Sensor Network

2.6 National Instruments Wireless Sensor Network Platform

The National Instruments Wireless Sensor Network (NI WSN) platform is an industrial wireless monitoring solution designed for reliable data aquisition and communication. The platform combines wireless sensor nodes and a centralized gateway to provide an integrated monitoring infrastructure.

The NI WSN platform is specifically designed to support distributed measurement systems requiring low power consumption, reliable wireless communication, and long-term operation.

The platform is based on the IEEE 802.15.4 wireless communication standard operating in the 2.4 GHz frequency band. This communication technology provides a balance between communication reliability, power efficiency, and network scalability.

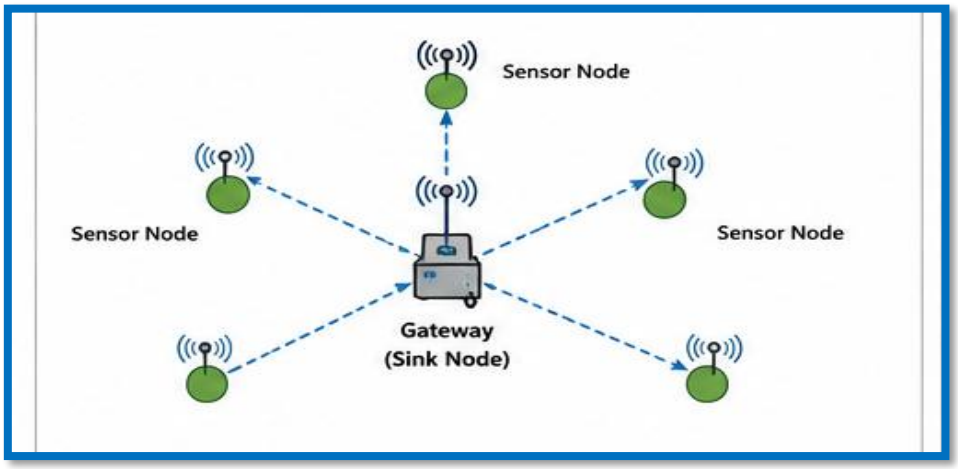


Figure 2.5: Star Topology in Wireless Sensor Networks

The NI WSN system consists of three major components:

Wireless Sensor Nodes

Ethernet Gateway

Wireless Communication Infrastructure

Wireless nodes acquire measurements and transmit information through the wireless network. The gateway receives the transmitted data and forwards it to external monitoring systems.

The NI WSN platform supports multiple wireless nodes within a single network, enabling monitoring over large geographical areas. The platform also incorporates power management mechanisms that allow battery-powered operation for extended periods.

Due to its industrial reliability, low-power design, and scalable architecture, the NI WSN platform has become a suitable solution for environmental monitoring, industrial measurement systems, research laboratories, and remote sensing applications.

The NI Wireless Sensor Network platform used in this project consists of the NI WSN-3202 Analog Input Node, the NI WSN-3212 Thermocouple Input Node, and the NI WSN-9791 Ethernet Gateway. The following sections provide a detailed discussion of each device and its role within the wireless monitoring system.

2.7 NI WSN-3202 Analog Input Node

Figure 2.6: WSN-3202 Analog Input Node

2.7.1 Introduction

The NI WSN-3202 is a wireless analog input node designed for remote measurement and monitoring applications. It combines high-resolution analog data acquisition with low-power wireless communication, enabling the deployment of distributed sensing systems without the need for extensive wiring. The node is specifically designed to operate as part of the National Instruments Wireless Sensor Network platform and communicates directly with the NI WSN-9791 Gateway.

The NI WSN-3202 is suitable for environmental monitoring, industrial automation, structural health monitoring, agricultural applications, and remote sensing systems. Its low power consumption and long battery life make it particularly useful in locations where continuous power sources may not be available.

2.7.2 Hardware Architecture

The NI WSN-3202 integrates several hardware components within a compact industrial enclosure. These components work together to perform data acquisition, signal processing, wireless communication, and power management.

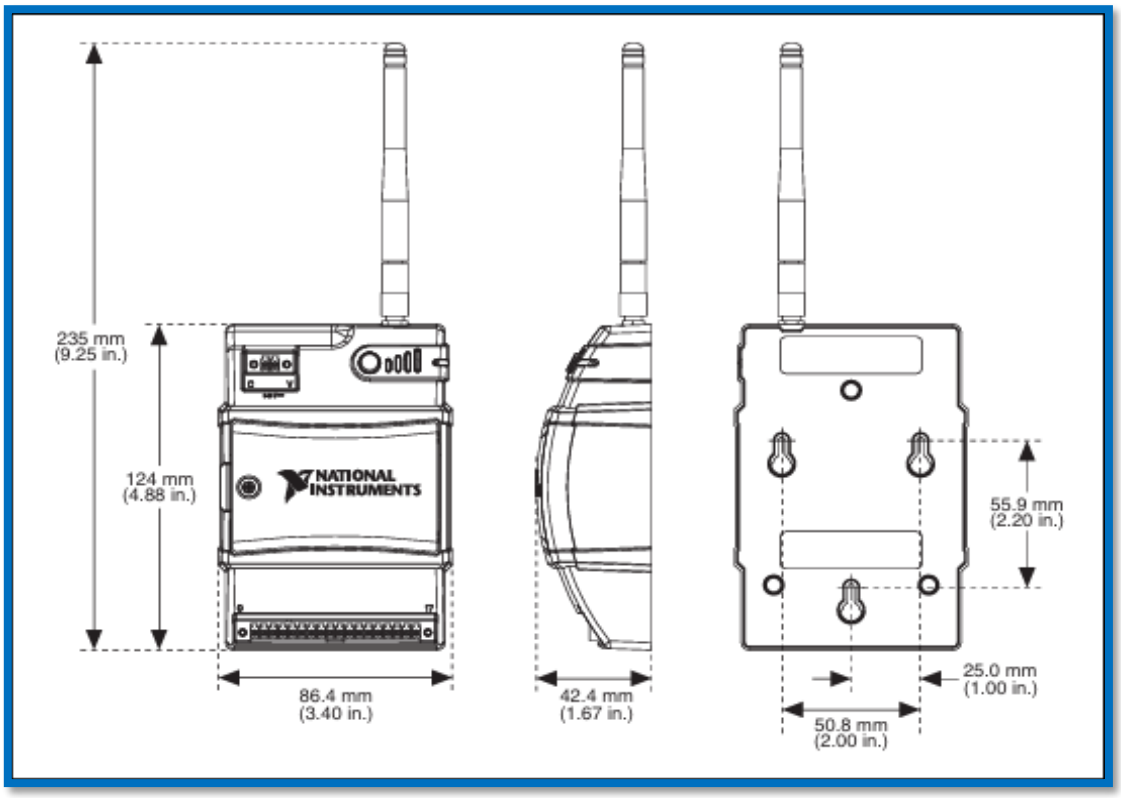


Figure 2.7: NI WSN-3202 Device Dimensions

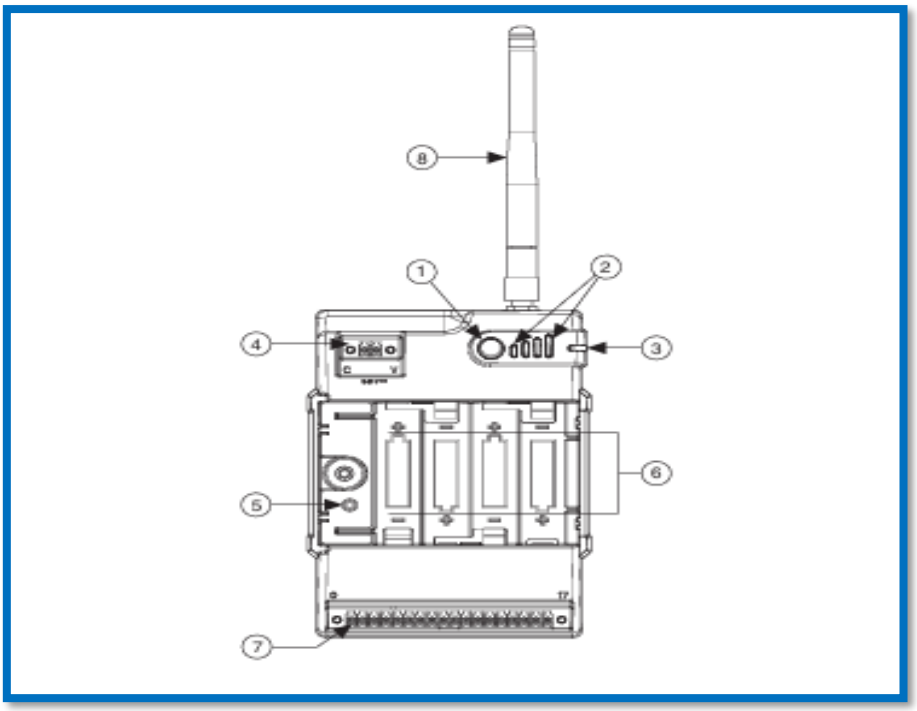


Figure 2.8: NI WSN-3202 Node Interface

1. **WSN Connect Button**
2. **Signal Strength/Status LEDs**
3. **User LED**
4. **9–30 V External Power Input**
5. **Reset Button**
6. **Battery Slots/Polarity Indicators**
7. **I/O Connector**
8. **Antenna**

The internal architecture consists of:

Analog Front-End Circuitry

16-Bit Analog-to-Digital Converter (ADC)

Low-Power Microcontroller

Flash Memory

IEEE 802.15.4 Wireless Radio

External Power Interface

Battery Backup System

Digital I/O Circuitry

The analog front-end receives input signals and conditions them before conversion into digital values. The microcontroller processes the acquired data and controls communication with the wireless network. The radio module transmits the measurements to the gateway using the IEEE 802.15.4 protocol.

2.7.3 Analog Input Characteristics

One of the primary features of the NI WSN-3202 is its analog measurement capability. The node provides four independent analog input channels that allow simultaneous monitoring of multiple measurement points.

Key Analog Input Specifications :

Table 2.1: WSN-3202 Analog Input Specifications

Parameter	*Value*
Number of Analog Inputs	*4 Channels*
ADC Resolution	*16 Bits*
Input Type	*Single-Ended*
Input Coupling	*DC*
Minimum Sample Interval	*1 Second*
Analog Bandwidth	*7 kHz*
Overvoltage Protection	*±30 V*

The high-resolution ADC allows precise measurement of analog signals while maintaining low power consumption.

2.7.4 Input Voltage Ranges

The NI WSN-3202 supports multiple programmable voltage ranges, allowing the user to select the most suitable measurement range depending on the application.

Supported Input Ranges

Voltage Range :

±10 V

±5 V

±2 V

±0.5 V

The NI WSN-3202 provides multiple programmable input voltage ranges, allowing the user to select the most appropriate measurement range for a specific sensor application. Choosing a smaller voltage range improves measurement resolution and overall accuracy.

2.7.5 Measurement Accuracy

Accurate data acquisition is critical for environmental monitoring applications. The NI WSN-3202 provides factory calibration and stores calibration coefficients in onboard EEPROM memory.

The following table summarizes the maximum measurement accuracy for different voltage ranges.

Table 2.2: WSN-3202 Analog Input Measurement Accuracy

Range	***Maximum Absolute Accuracy***
±10 V	*38.21 mV*
±5 V	*22.91 mV*
±2 V	*10.52 mV*
±0.5 V	*2.69 mV*

The measurement accuracy improves as the selected voltage range decreases, making the device suitable for both large and small signal measurements.

2.7.6 Wireless Communication Characteristics

Wireless communication is achieved using the IEEE 802.15.4 standard operating in the Industrial, Scientific, and Medical (ISM) frequency band.

Communication Specifications :

Table 2.3: WSN-3202 Wireless Communication Specifications

Parameter	***Value***
Wireless Standard	*IEEE 802.15.4*
Frequency Band	*2.4 GHz*
Data Rate	*250 kbps*
Channels	*11 – 24*
Modulation	*DSSS (O-QPSK)*
Receiver Sensitivity	*-102 dBm*

The selected communication protocol is optimized for low-power operation while maintaining reliable wireless connectivity.

2.7.7 Communication Range

Communication distance depends on environmental conditions and regional transmission regulations.

Wireless Range:

Table 2.4 WSN-3202 Wireless Range

Region	***Maximum Range***
Americas	*Up to 300 m*
International	*Up to 150 m*

These communication distances make the NI WSN-3202 suitable for deployment in large monitoring areas.

2.7.8 Antenna System

The wireless communication subsystem uses an omnidirectional antenna to provide full coverage around the node

Antenna Characteristics :

Table 2.5: WSN-3202 Antenna Characteristics

Parameter	***Value***
Connector Type	*RP-SMA*
Impedance	*50 Ω*
Gain	*1.5 dBi*
Directivity	*Omnidirectional*

The antenna allows reliable communication with the gateway regardless of the orientation of the node.

2.7.9 Power Supply Options

The NI WSN-3202 supports two power supply methods:

1- Internal Battery Power

The node can operate using four AA alkaline batteries.

Recommended battery types include:

Energizer EN91 AA

Duracell MN1500 AA

Battery voltage range: 3.6 V to 7.5 V

2-External Power Supply

External Voltage Range : 9 V to 30 V DC

When both battery and external power are connected, the device automatically prioritizes the external source while maintaining batteries as backup power.

2.7.10 Power Consumption

Power consumption depends on the sampling interval selected for the node.

Battery Operation

Table 2.6: WSN-3202 Power Consumption under Battery Operation

Sampling Interval	*Power Consumption*
60 Seconds	*0.5 mW*
1 Second	*13.3 mW*

External Power Operation

Table 2.7: WSN-3202 Power Consumption under External Power Operation

Sampling Interval	*Power Consumption*
60 Seconds	*16 mW*
1 Second	*33 mW*

The low power consumption significantly extends the operating lifetime of the device.

2.7.11 Battery Lifetime

Battery life is one of the most important characteristics of wireless sensor networks because it directly affects maintenance requirements.

Expected Battery Lifetime

Table 2.8: Expected Battery Lifetime of the WSN-3202

Sampling Interval	***Battery Life***
60 Seconds	*Up to 3 Years*
1 Second	*Up to 1 Month*

The ability to operate for several years on standard AA batteries demonstrates the efficiency of the NI WSN platform.

2.7.12 Digital Input and Output Features

In addition to analog inputs, the NI WSN-3202 includes four configurable digital channels.

The digital channels can operate in several modes:

Drive High Only

Drive Low Only

Drive High and Low

Tristate Mode

These capabilities allow the node to interface with external devices and control systems when required.

2.7.13 Environmental Specifications

The NI WSN-3202 is designed for operation in demanding industrial and outdoor environments.

Operating Conditions

Table 2.9: WSN-3202 Environmental Specification

Parameter	***Value***
Operating Temperature	*-40°C to +70°C*
Storage Temperature	*-40°C to +85°C*
Operating Humidity	*10% – 90% RH*
Storage Humidity	*5% – 95% RH*
Maximum Altitude	*2000 m*

These specifications ensure reliable operation under a wide range of environmental conditions.

2.7.14 Physical Characteristics

The compact design of the NI WSN-3202 allows easy installation in industrial and field applications.

Table 2.10: WSN-3202 Physical Characteristics

Parameter	***Value***
Dimensions	*124 × 86.4 × 42.4 mm*
Weight	*Approximately 242 g*
Weight with Antenna	*Approximately 256 g*

The device can be mounted using DIN rails, panel mounts, or integrated mounting slots depending on installation requirements.

2.7.15 Advantages of NI WSN-3202

The NI WSN-3202 offers several advantages for wireless monitoring applications:

High-resolution analog measurements

Low power consumption

Long battery lifetime

Reliable wireless communication

Industrial-grade design

Flexible power options

Easy network integration

Large communication range

Support for distributed sensing systems

These characteristics make the NI WSN-3202 an effective solution for remote monitoring applications and wireless data acquisition systems.

2.8 NI WSN-3212 Thermocouple Node

Figure 2.9: NI WSN-3212 Thermocouple Node

2.8.1 Introduction

The NI WSN-3212 is a wireless thermocouple input node designed for remote temperature measurement applications within the National Instruments Wireless Sensor Network platform.

The device combines high-accuracy temperature acquisition with low-power wireless communication, allowing temperature measurements to be collected from distributed locations and transmitted wirelessly to a central gateway.

The NI WSN-3212 is particularly suitable for industrial monitoring, environmental observation, energy management systems, process control, and remote measurement applications where wired installation may be difficult or expensive.

By integrating thermocouple measurement circuitry, signal conditioning, and wireless communication into a single device, the node provides a complete solution for distributed temperature monitoring.

2.8.2 Device Overview

The NI WSN-3212 functions as an intelligent wireless measurement node capable of acquiring temperature data and transmitting it through the wireless sensor network.

The node communicates directly with the NI WSN-9791 Gateway using IEEE 802.15.4 wireless communication technology.

The device is designed to operate with minimal power consumption while maintaining high measurement accuracy.

This makes it suitable for long-term deployment in remote locations where frequent maintenance is undesirable.

The node can be powered using either internal batteries or an external DC power source, providing flexibility in installation and operation.

2.8.3 Hardware Architecture

The NI WSN-3212 integrates several hardware subsystems that work together to perform measurement and communication tasks.

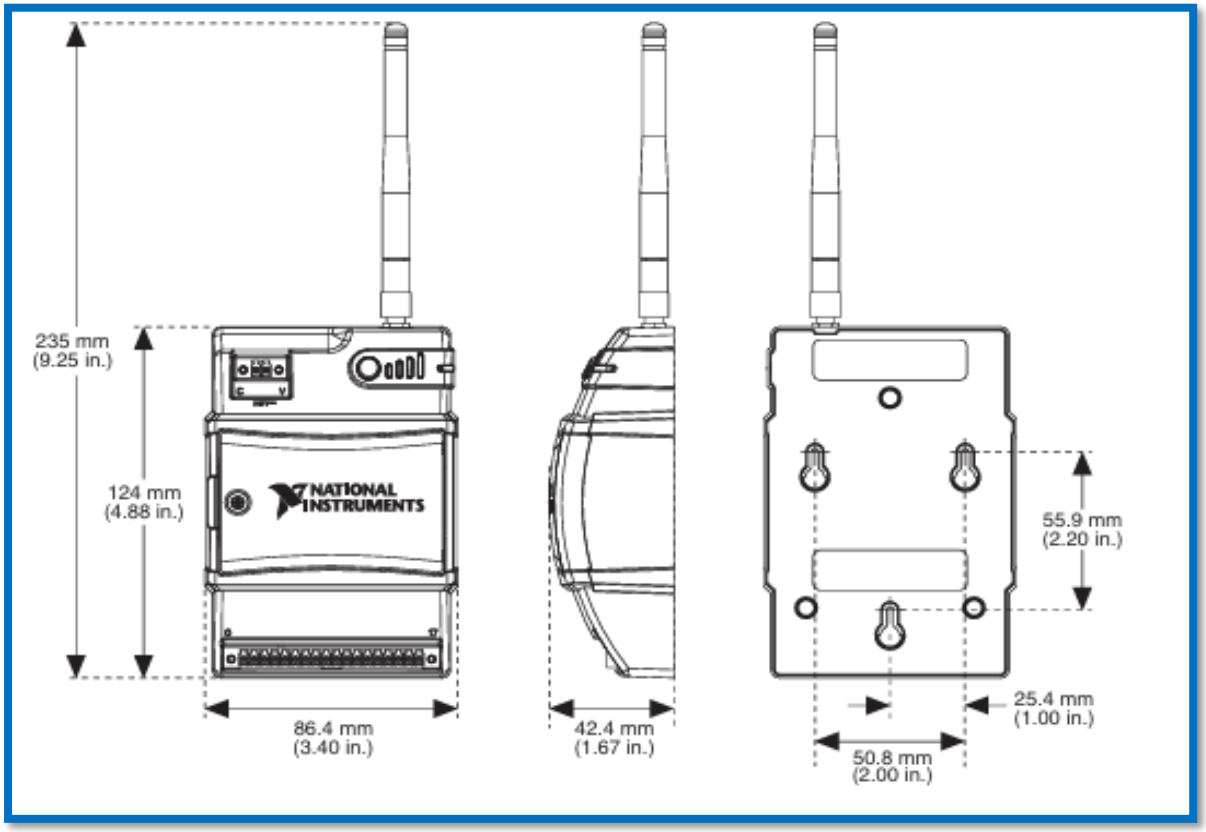


Figure 2.10: NI WSN-3212 Device Dimension

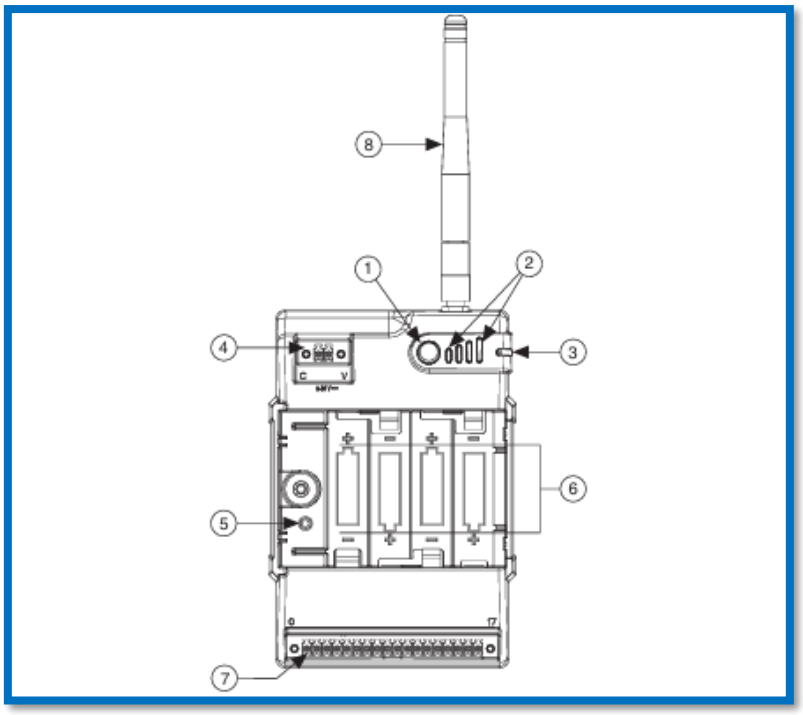


Figure 2.11: NI WSN-3212 Node Interface

1. **WSN Connect Button**
2. **Signal Strength/Status LEDs**
3. **User LED**
4. **9–30 V External Power Input**
5. **Reset Button**
6. **Battery Slots/Polarity Indicators**
7. **I/O Connector**
8. **Antenna**

The main internal components include:

Thermocouple Input Circuitry

High-Resolution Analog-to-Digital Converter

Low-Power Microcontroller

Embedded Flash Memory

IEEE 802.15.4 Radio Module

Battery Management System

External Power Interface

Wireless Communication Controller

The thermocouple input circuitry performs signal conditioning and cold-junction compensation before converting the temperature signal into digital data. The microcontroller processes the measurements and controls wireless transmission to the gateway.

2.8.4 Thermocouple Measurement Capability

The primary function of the NI WSN-3212 is accurate temperature acquisition using thermocouple sensors.

The device provides:

Four thermocouple input channels

High-resolution temperature measurements

Cold Junction Compensation (CJC)

Continuous monitoring capability

Wireless transmission of temperature data

The use of thermocouple technology allows temperature monitoring across a wide operating range while maintaining measurement reliability.

2.8.5 High-Resolution Data Acquisition

The NI WSN-3212 utilizes a high-resolution analog-to-digital conversion system to ensure accurate temperature measurements.

Measurement Characteristics

***Table 2.11*: *WSN-3212 Thermocouple Measurement* Characteristics**

Parameter	***Specification***
Number of Inputs	*4 Channels*
Measurement Type	*Thermocouple*
Resolution	*24-bit*
Input Configuration	*Differential*
Sampling Capability	*Periodic Measurement*

The 24-bit measurement resolution allows detection of very small temperature changes, improving monitoring accuracy in environmental and industrial applications.

2.8.6 Wireless Communication System

The NI WSN-3212 communicates with the gateway through a low-power wireless communication link based on the IEEE 802.15.4 standard.

Wireless Specifications

Table 2.12: WSN-3212 Wireless Communication Specifications

Parameter	***Specification***
Communication Standard	*IEEE 802.15.4*
Frequency Band	*2.4 GHz ISM*
Data Rate	*250 kbps*
Communication Type	*Wireless*
Network Integration	*NI WSN Platform*

The communication system is optimized for low power consumption while maintaining reliable data transfer.

2.8.7 Communication Range

The wireless communication range depends on environmental conditions, antenna placement, and local regulations.

Typical communication distances include:

Table 2.13: WSN-3212 Wireless Communication Range

Region	***Maximum Range***
Americas	*Up to 300 m*
International	*Up to 150 m*

These distances allow deployment of temperature monitoring systems over wide geographical areas.

2.8.8 Power Supply Options

The NI WSN-3212 supports multiple power supply methods, allowing flexibility in deployment.

1- Internal Battery Operation

The node can operate using four AA alkaline batteries.

Benefits include:

Portable installation

Reduced wiring requirements

Simplified deployment

Backup power capability

2-External Power Operation

The device also supports external DC power sources.

External power operation is recommended for applications requiring frequent measurements or continuous operation.

2.8.9 Low-Power Operation

One of the most important characteristics of the NI WSN-3212 is its low-power design.

The node spends most of its operating time in a low-power sleep state and becomes active only during measurement and communication intervals. This approach significantly reduces power consumption and extends battery lifetime.

Low-power operation is particularly important in environmental monitoring systems where devices may remain deployed for extended periods without maintenance.

2.8.10 Battery Lifetime

Battery life depends largely on the selected sampling interval and communication frequency.

By increasing the sampling interval, the device can remain in sleep mode for longer periods, thereby reducing energy consumption.

The efficient power management architecture allows long-term operation using standard AA batteries, making the NI WSN-3212 suitable for remote monitoring applications.

2.8.11 Environmental Specifications

The NI WSN-3212 is designed to operate under challenging environmental conditions.

Operating Conditions

Table 2.14: WSN-3212 Environmental Specifications

Parameter	***Value***
Operating Temperature	*-40°C to +70°C*
Storage Temperature	*-40°C to +85°C*
Operating Humidity	*Up to 90% RH*
Storage Humidity	*Up to 95% RH*
Maximum Altitude	*2000 m*

These specifications enable reliable operation in industrial and outdoor environments.

2.8.12 Mechanical Characteristics

The device is designed for industrial deployment and can be mounted using several installation methods.

Mounting Options

DIN Rail Mounting

Panel Mounting

Industrial Enclosures

The compact structure simplifies installation while maintaining mechanical robustness.

2.8.13 Network Integration

The NI WSN-3212 is fully integrated within the National Instruments Wireless Sensor Network architecture.

The integration process involves:

1. Connecting the node to the wireless network.
2. Associating the node with the NI WSN-9791 Gateway.
3. Assigning a unique node identifier.
4. Establishing wireless communication.
5. Beginning measurement transmission.

This architecture enables scalable deployment of multiple wireless nodes within the same monitoring system.

2.8.14 Advantages of NI WSN-3212

The NI WSN-3212 offers several advantages that make it suitable for distributed temperature monitoring applications:

High-resolution temperature measurement

Wireless communication capability

Low power consumption

Long battery life

Industrial-grade reliability

Flexible installation options

Wide environmental operating range

Easy integration into WSN systems

Scalable network deployment

These characteristics make the NI WSN-3212 a powerful solution for remote temperature monitoring in industrial and environmental applications.

2.9 NI WSN-9791 Ethernet Gateway

Figure 2.12: NI WSN-9791 Ethernet Gateway

2.9.1 Introduction

The NI WSN-9791 Ethernet Gateway is the central communication device within the National Instruments Wireless Sensor Network platform. It serves as the coordinator of the wireless network and provides communication between wireless sensor nodes and external monitoring systems. The gateway collects measurement data from all connected wireless nodes and transfers this information to the host computer through an Ethernet connection.

As the core component of the network, the gateway manages wireless communication, synchronizes network activities, and ensures reliable data transmission between distributed sensor nodes and monitoring applications. Its ability to support multiple wireless nodes makes it suitable for large-scale monitoring systems requiring centralized data collection and management.

2.9.2 Gateway Overview

The NI WSN-9791 functions as the central controller of the wireless sensor network. It establishes communication links with all connected wireless nodes and coordinates the exchange of information within the network.

The gateway creates a bridge between two communication environments:

Wireless Sensor Network (IEEE 802.15.4)

Ethernet-Based Monitoring Network

This dual communication capability enables seamless integration between wireless measurement systems and external computing platforms.

The gateway continuously monitors network status and manages communication with all connected nodes to ensure reliable operation.

2.9.3 Hardware Architecture

The NI WSN-9791 consists of several integrated subsystems designed to support wireless communication and network management.

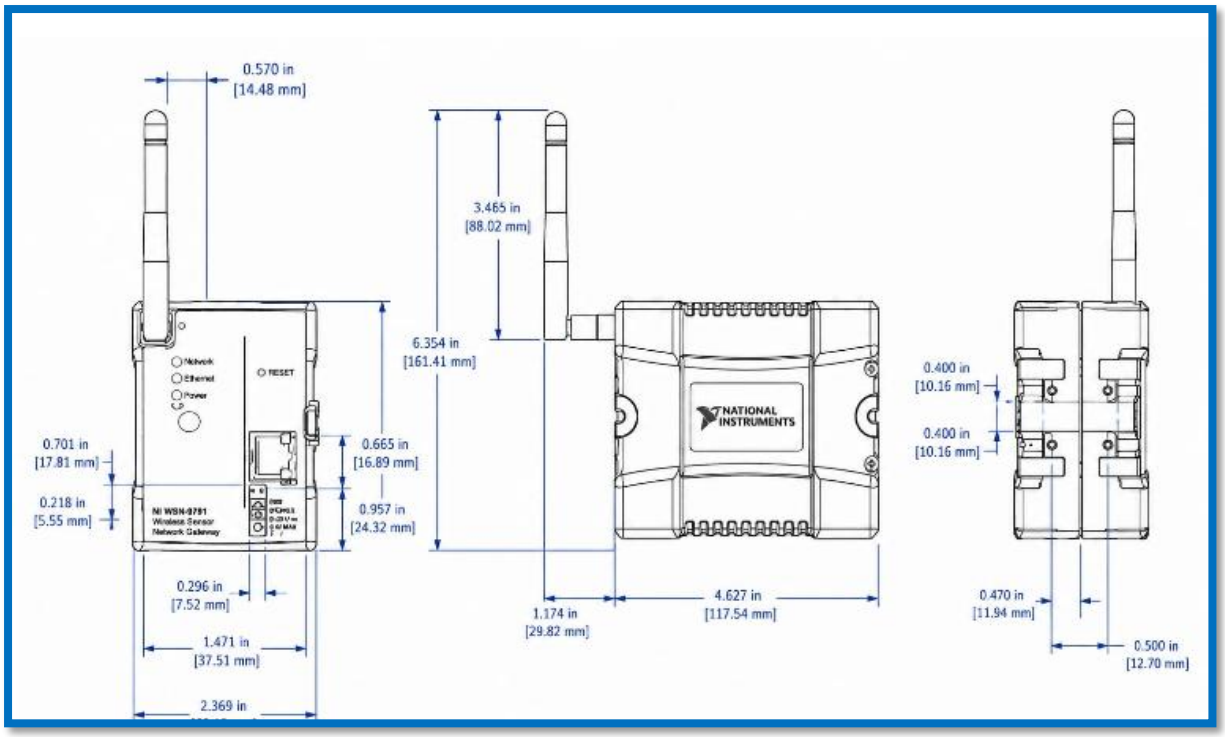


Figure 2.13: WSN-9791 Ethernet Gateway Dimension

The main hardware components include:

Embedded Processor

IEEE 802.15.4 Wireless Radio

Ethernet Communication Interface

Flash Memory

RAM Memory

Power Supply Circuitry

Status Indicators

Network Management Controller

The embedded processor executes network management functions, while the wireless radio communicates with sensor nodes. The Ethernet interface transfers collected information to external systems.

This architecture enables the gateway to act as a communication bridge between wireless and wired networks.

2.9.4 Network Coordination Functions

One of the most important responsibilities of the gateway is network coordination.

The gateway performs the following tasks:

Establishing wireless communication links

Registering wireless sensor nodes

Managing data transmission schedules

Monitoring node status

Synchronizing network communication

Maintaining network reliability

Handling communication requests

By coordinating these activities, the gateway ensures efficient network operation and prevents communication conflicts among multiple nodes.

2.9.5 Wireless Communication Technology

The NI WSN-9791 uses the IEEE 802.15.4 wireless communication protocol, which is specifically designed for low-power wireless sensor networks.

Communication Characteristics

Table 2.15: WSN-9791 Wireless Communication Specification

Parameter	***Specification***
Communication Standard	*IEEE 802.15.4*
Frequency Band	*2.4 GHz*
Communication Type	*Wireless*
Network Architecture	*Star Topology*
Maximum Supported Nodes	*36 Nodes*

The IEEE 802.15.4 protocol was selected because it provides reliable communication while minimizing power consumption in wireless devices.

2.9.6 Ethernet Connectivity

The gateway connects to the host computer through a standard Ethernet interface.

The Ethernet connection offers several advantages:

High communication reliability

Fast data transfer

Easy integration with existing networks

Remote accessibility

Long communication distance

The Ethernet interface enables sensor data to be transferred from the wireless network to monitoring systems, databases, and control applications.

2.9.7 Network Capacity

The NI WSN-9791 is designed to support multiple wireless nodes within a single network.

Network Capacity

Table 2.15: WSN-9791 Network Capacity Specifications

Parameter	***Value***
Maximum Number of Nodes	*36*
Communication Type	*Point-to-Gateway*
Network Structure	*Star Topology*

This capacity allows the deployment of large monitoring systems covering extensive geographical areas.

As additional nodes are added, the gateway automatically manages communication and maintains network synchronization.

2.9.8 Node Registration Process

Before communication can begin, each wireless node must be registered with the gateway.

The registration process typically involves:

1. Powering the gateway.
2. Activating the wireless node.
3. Assigning a unique node identifier.
4. Establishing a wireless communication link.
5. Verifying successful connection.

Once registered, the node becomes part of the wireless sensor network and can begin transmitting measurement data.

The gateway maintains information about all connected nodes and continuously monitors their operational status.

2.9.9 Data Flow within the Network

The NI WSN-9791 acts as the central collection point for all network traffic.

The communication sequence can be summarized as follows:

1. A wireless node acquires measurement data.

2. The node processes the acquired information.

3. The data are transmitted wirelessly to the gateway.

4. The gateway receives and verifies the transmitted information.

5. The data are forwarded through the Ethernet interface.

6. External systems receive and process the information.

This architecture provides a reliable mechanism for collecting measurements from distributed locations.

2.9.10 Status Monitoring and Diagnostics

The NI WSN-9791 includes built-in diagnostic and monitoring capabilities.

The gateway continuously monitors:

Node communication status

Network connectivity

Communication quality

Wireless signal strength

Device health information

Diagnostic information helps users identify communication issues and maintain reliable network operation.

2.9.11 Reliability and Industrial Performance

The gateway is designed for industrial monitoring applications where reliability is critical.

Several features contribute to reliable operation:

Robust communication protocols

Automatic node management

Error detection mechanisms

Continuous network supervision

Stable Ethernet communication

These features ensure consistent data transmission even in demanding environments.

2.9.12 Security Considerations

Wireless sensor networks require protection against unauthorized access and communication interference.

The NI WSN platform incorporates security mechanisms that help protect network communication and prevent unauthorized devices from joining the network.

Security measures improve data integrity and increase system reliability in industrial applications.

2.9.13 Advantages of the NI WSN-9791 Gateway

The NI WSN-9791 offers several benefits for wireless monitoring systems:

Centralized network management

Support for multiple wireless nodes

Reliable communication architecture

Ethernet connectivity

Low-power wireless networking

Scalable deployment

Industrial-grade performance

Easy network expansion

Remote monitoring capability

These advantages make the gateway a key component of the National Instruments Wireless Sensor Network platform.

2.9.14 Applications of the NI WSN-9791 Gateway

The gateway can be utilized in a wide range of monitoring applications, including:

Environmental monitoring systems

Industrial automation

Energy management systems

Structural health monitoring

Smart infrastructure

Remote sensing networks

Research and development applications

Its flexibility allows deployment in both industrial and academic projects.

2.10 Wireless Sensor Network Topology

2.10.1 Introduction

Network topology refers to the physical and logical arrangement of devices within a communication network. The topology determines how sensor nodes communicate with each other and with the gateway.

The choice of topology significantly influences communication reliability, network complexity, scalability, and power consumption.

2.10.2 Star Topology

The National Instruments Wireless Sensor Network primarily utilizes a star topology configuration.

In a star topology, all sensor nodes communicate directly with a central gateway rather than communicating through intermediate nodes.

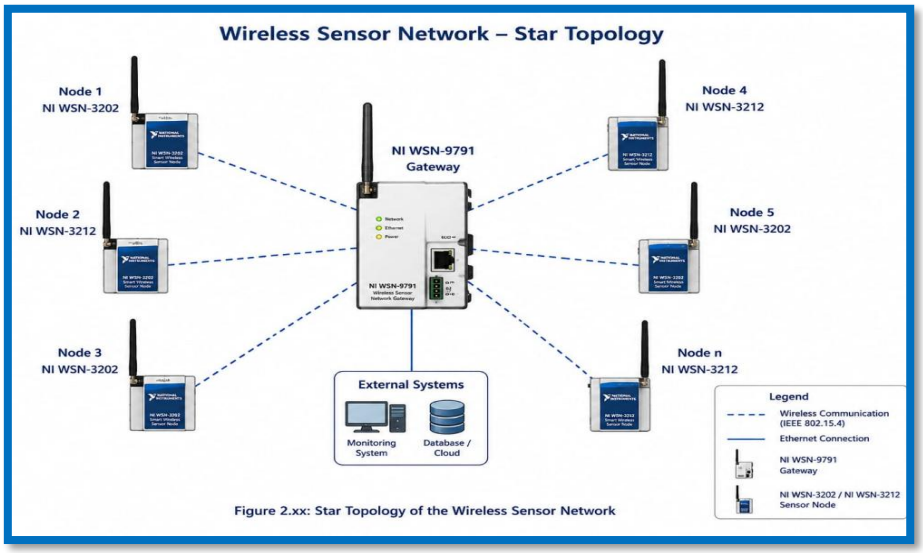


Figure 2.14: Star Topology of the NI Wireless Sensor Network

Structure of Star Topology

In the NI Wireless Sensor Network, the WSN-9791 Gateway serves as the central communication point.

Each wireless sensor node (such as the NI WSN-3202 and NI WSN-3212) communicates directly with the gateway using the IEEE 802.15.4 protocol. The gateway collects data from all connected nodes and forwards it to the monitoring computer through the Ethernet interface for processing, visualization, and storage.

2.10.3 Operation of Star Networks

The operation of a star network follows a straight forward communication model:

1. Sensor nodes collect measurements.
2. The measurements are processed locally.
3. Data are transmitted directly to the gateway.
4. The gateway receives the information.
5. The gateway forwards the data to external systems.

Because all communication passes through the gateway, network management becomes significantly simpler.

2.10.4 Advantages of Star Topology

Several advantages make star topology suitable for wireless monitoring systems

Simplicity

The network structure is straightforward and easy to implement.

Easy Maintenance

Individual nodes can be added or removed without affecting the operation of other nodes.

Centralized Management

The gateway manages all communication activities.

Improved Reliability

Communication paths are clearly defined and easier to monitor.

Scalability

Additional nodes can be incorporated into the network with minimal configuration changes.

2.10.5 Limitations of Star Topology

Despite its advantages, star topology also has several limitations.

1-Single Point of Failure

The gateway represents a critical component of the network. If the gateway fails, communication with all nodes is interrupted.

2- Range Restrictions

All nodes must be located within communication range of the gateway.

3-Gateway Workload

As the number of nodes increases, the communication workload on the gateway also increases.

2.11 Power Management in Wireless Sensor Networks

2.11.1 Introduction

Power consumption is one of the most important design considerations in Wireless Sensor Networks.

Since many wireless nodes operate using batteries, efficient energy management is essential for maximizing operational lifetime and minimizing maintenance requirements.

The National Instruments Wireless Sensor Network platform is specifically designed to provide long battery life while maintaining reliable communication and measurement performance.

2.11.2 Energy Consumption Sources

The primary sources of power consumption within a wireless sensor node include:

Sensor measurements

Analog-to-digital conversion

Data processing

Wireless transmission

Wireless reception

Memory operations

Among these activities, wireless communication typically consumes the largest amount of energy.

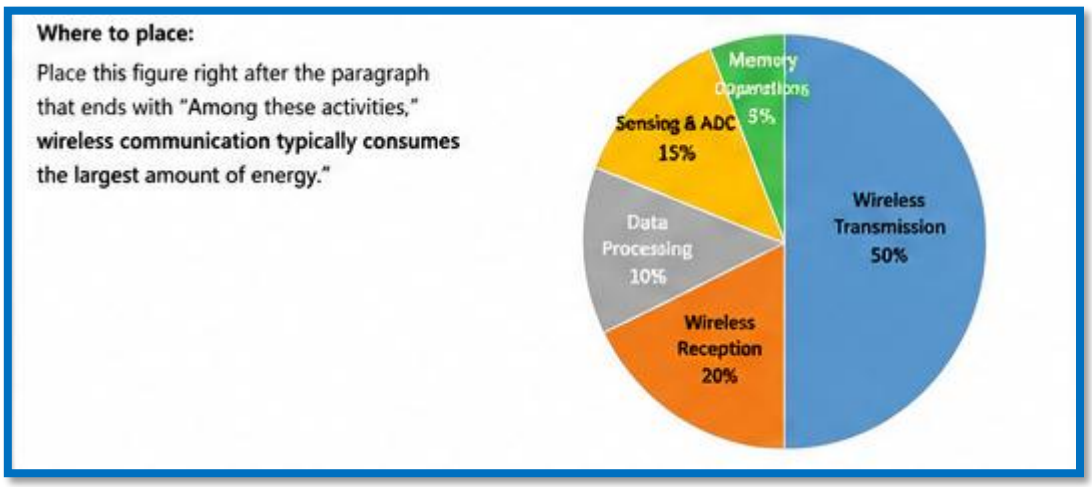

Where to place:
Place this figure right after the paragraph that ends with "Among these activities," wireless communication typically consumes the largest amount of energy."
Memory Operations 5%
Sensing & ADC 15%
Wireless Transmission 50%
Data Processing 10%
Wireless Reception 20%

Figure 2.15: Main Sources of Energy Consumption in a Wireless Sensor Node

2.11.3 Sleep Mode Operation

To reduce energy consumption, wireless nodes spend most of their operating time in sleep mode.

During sleep mode:

The processor operates at minimum power.

The radio transmitter is disabled.

Unnecessary circuitry is turned off.

The node wakes up only when measurements or communication tasks are required.

This strategy significantly extends battery lifetime.

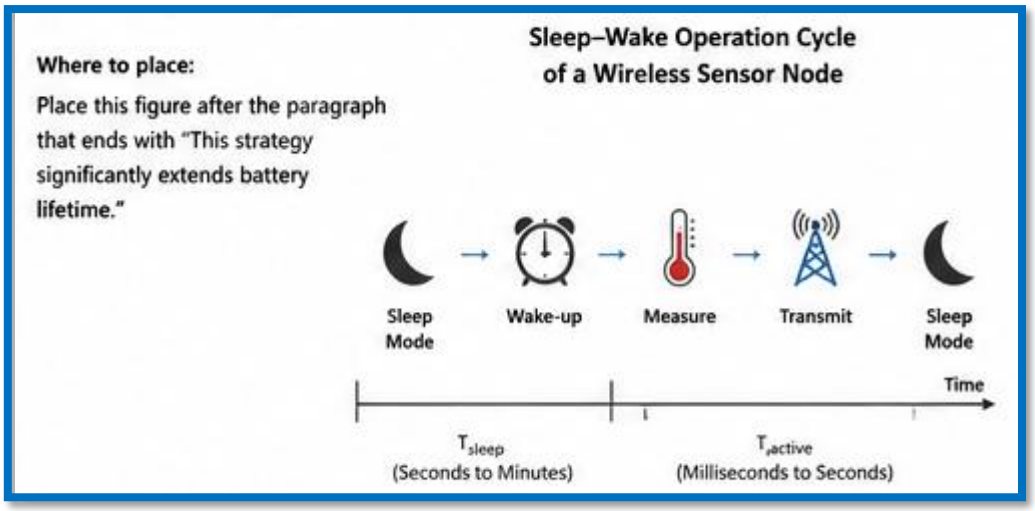


Figure 2.16: Sleep–Wake Operation Cycle of a Wireless Sensor Node

2.11.4 Battery Lifetime Considerations

Several factors influence battery lifetime in wireless sensor networks:

Sampling interval

Transmission frequency

Communication distance

Environmental conditions

Battery type

Number of measurements

Increasing the sampling interval generally reduces energy consumption and extends operational lifetime.

For this reason, many environmental monitoring applications utilize sampling intervals ranging from several seconds to several minutes.

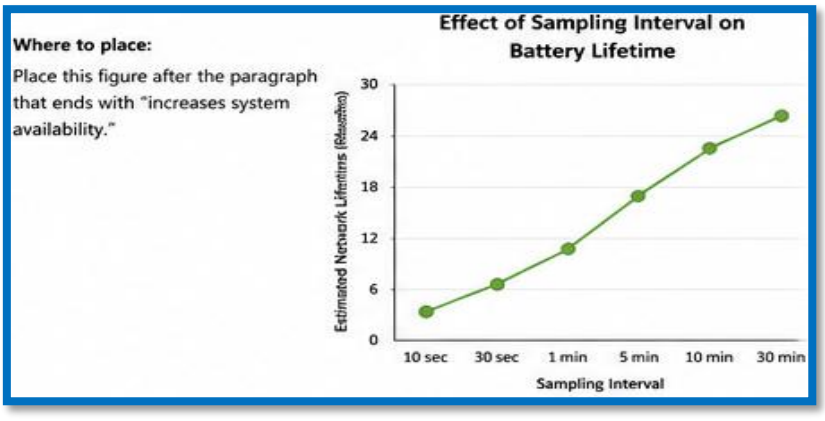


Figure 2.17: Effect of Sampling Interval on Battery Lifetime

2.11.5 Importance of Energy Efficiency

Energy-efficient operation provides several benefits:

Reduced maintenance costs

Extended deployment periods

Increased network reliability

Improved scalability

Lower operating expenses

Consequently, energy management remains one of the key design objectives in modern Wireless Sensor Networks.

2.12 Wireless Sensor Network Performance Metrics

2.12.1 Introduction

The performance of a Wireless Sensor Network is evaluated using several metrics that determine its efficiency, reliability, scalability, and energy consumption. These metrics are essential for assessing whether the network can satisfy the requirements of a particular monitoring application.

In environmental monitoring systems, performance metrics help determine the effectiveness of data acquisition and communication processes under different operating conditions.

2.12.2 Network Lifetime

Network lifetime is one of the most important performance indicators in wireless sensor networks. It represents the period during which the network can continue operating before battery depletion causes node failures.

Several factors influence network lifetime:

Battery capacity

Sampling interval

Communication frequency

Transmission power

Environmental conditions

Maximizing network lifetime is a major design objective because it reduces maintenance costs and increases system availability.

2.12.3 Communication Reliability

Communication reliability refers to the ability of the network to deliver data successfully from sensor nodes to the gateway.

Reliable communication is critical because monitoring systems depend on accurate and timely information.

Factors affecting reliability include:

Signal strength

Interference

Distance between devices

Obstacles in the communication path

Environmental conditions

A reliable network minimizes packet loss and ensures accurate monitoring.

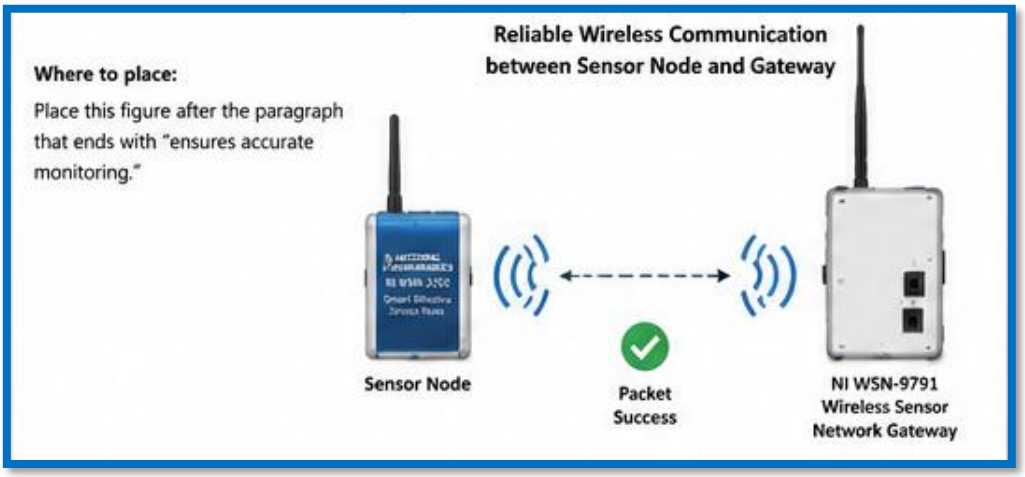


*Figure 2.18: **Reliable Wireless Communication between Sensor Node and Gateway***

2.12.4 Scalability

Scalability refers to the ability of a wireless sensor network to accommodate additional nodes without significant degradation in performance.

A scalable network allows future expansion while maintaining reliable communication and efficient operation.

The NI Wireless Sensor Network supports multiple nodes through centralized gateway management, enabling flexible deployment and network growth.

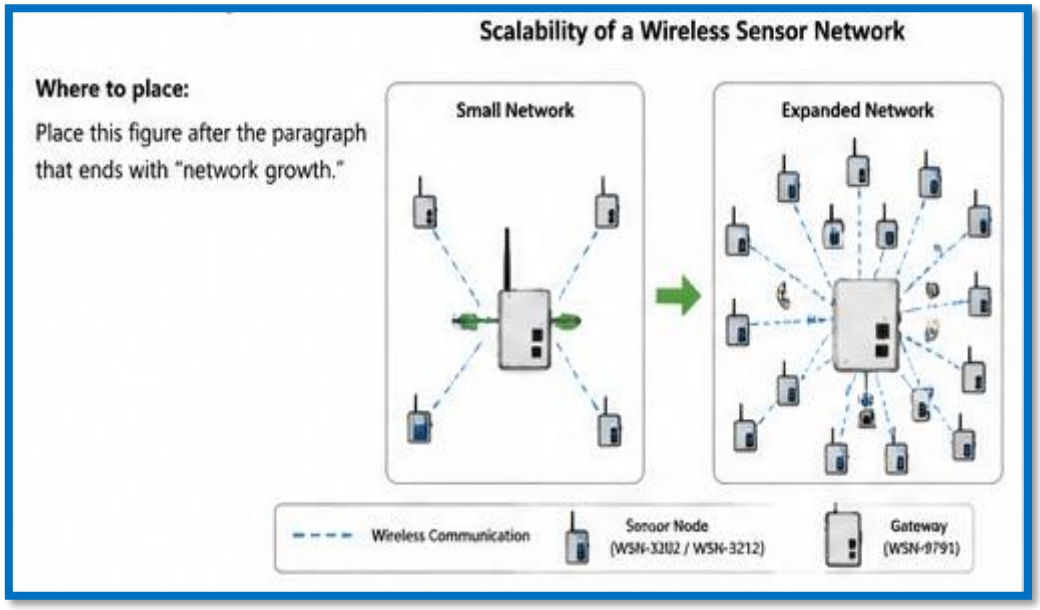


Figure 2.19: Scalability of a Wireless Sensor Network

2.12.5 Latency

Latency is the time required for data to travel from a wireless node to the gateway.

Low latency is desirable in applications requiring immediate response to changing environmental conditions.

Factors influencing latency include:

Sampling interval

Processing time

Communication protocol

Network traffic

The latency requirements vary depending on the monitoring application.

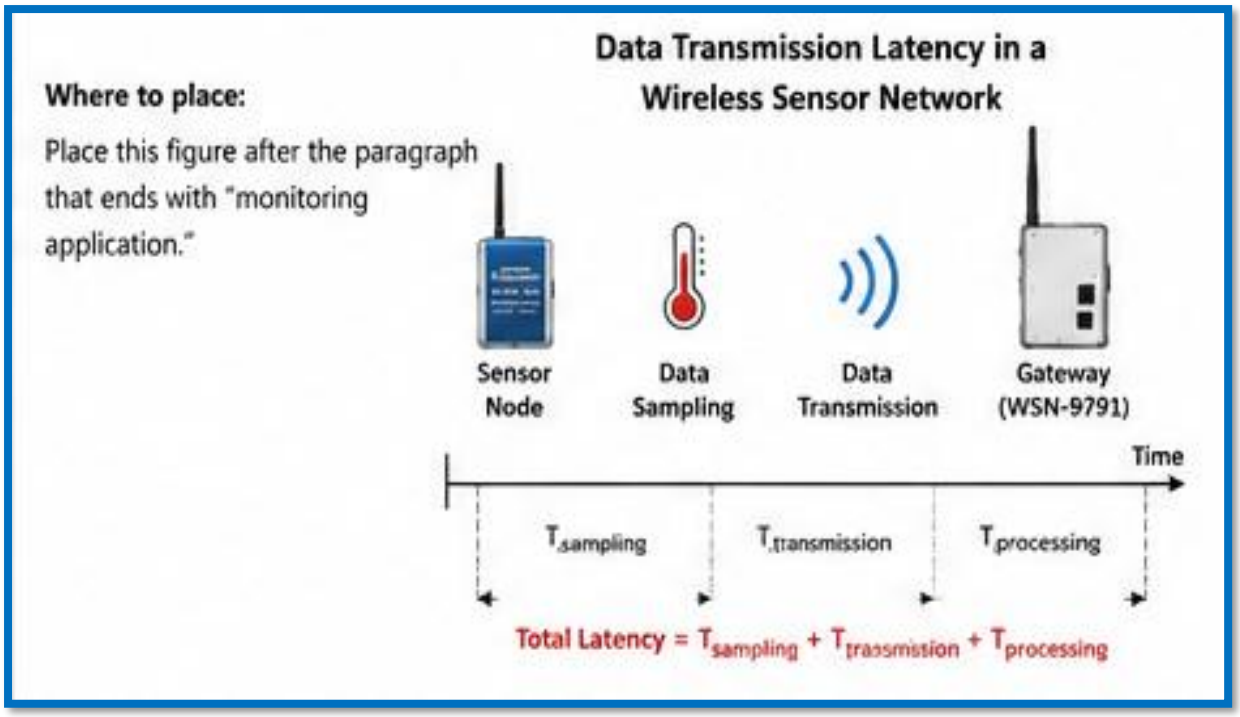

Where to place:
Place this figure after the paragraph that ends with "monitoring application."
Data Transmission Latency in a Wireless Sensor Network
Sensor Node
Data Sampling
Data Transmission
Gateway (WSN-9791)
Time
T sampling
T transmission
T processing
Total Latency = T sampling + T transmision + T processing

Figure 2.20: Data Transmission Latency in a Wireless Sensor Network

2.12.6 Throughput

Throughput represents the amount of data successfully transmitted through the network over a given period.

Although wireless sensor networks generally operate with relatively low data rates, adequate throughput remains important for supporting multiple sensor nodes and continuous monitoring activities.

2.13 Advantages of Wireless Sensor Networks in Environmental Monitoring

2.13.1 Introduction

Environmental monitoring applications often involve large geographical areas where conventional wired monitoring systems are difficult to deploy. Wireless Sensor Networks provide an effective solution by enabling distributed sensing and wireless communication.

The adoption of WSN technology has significantly improved the efficiency of environmental monitoring systems.

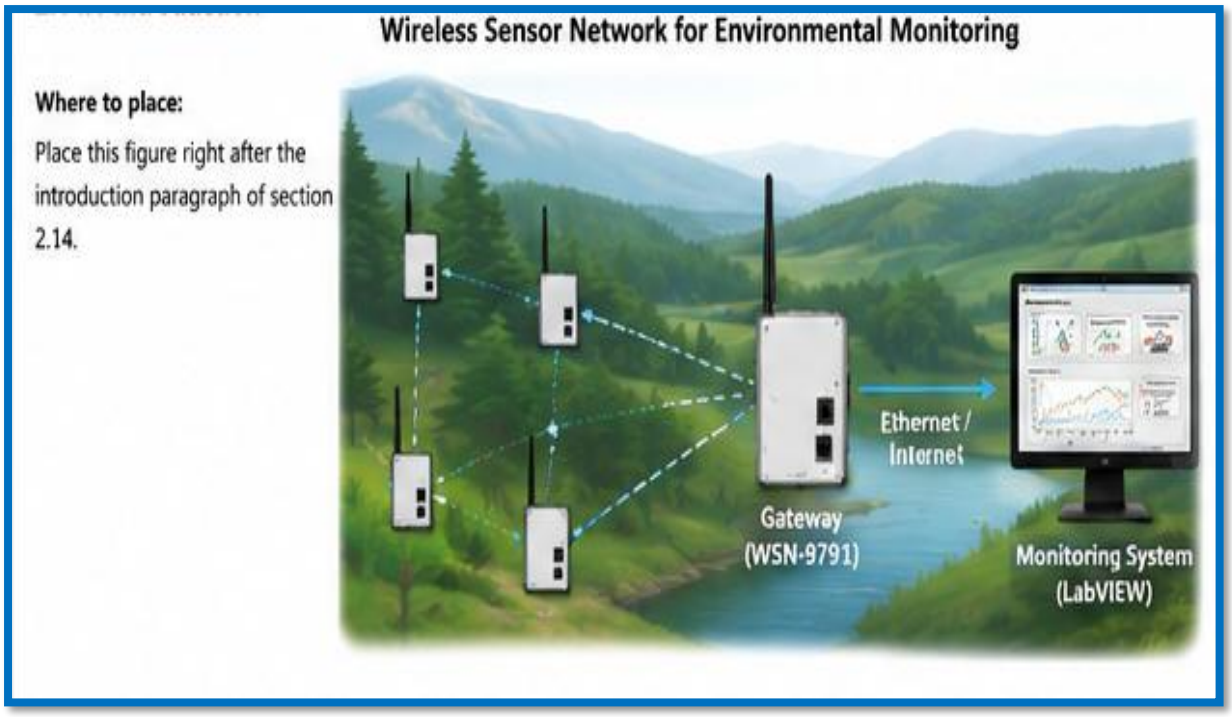

Wireless Sensor Network for Environmental Monitoring
Where to place:
Place this figure right after the introduction paragraph of section 2.14.
Ethernet / Internet
Gateway (WSN-9791)
Monitoring System (LabVIEW)

Figure 2.21: Wireless Sensor Network for Environmental Monitoring

2.13.2 Reduced Installation Cost

Traditional wired monitoring systems require extensive cabling infrastructure, increasing installation complexity and cost.

Wireless sensor networks eliminate much of this wiring, reducing both material and labor expenses.

2.13.3 Flexible Deployment

Wireless nodes can be installed in remote or difficult-to-access locations without requiring physical communication cables.

This flexibility simplifies system deployment and allows monitoring of large areas.

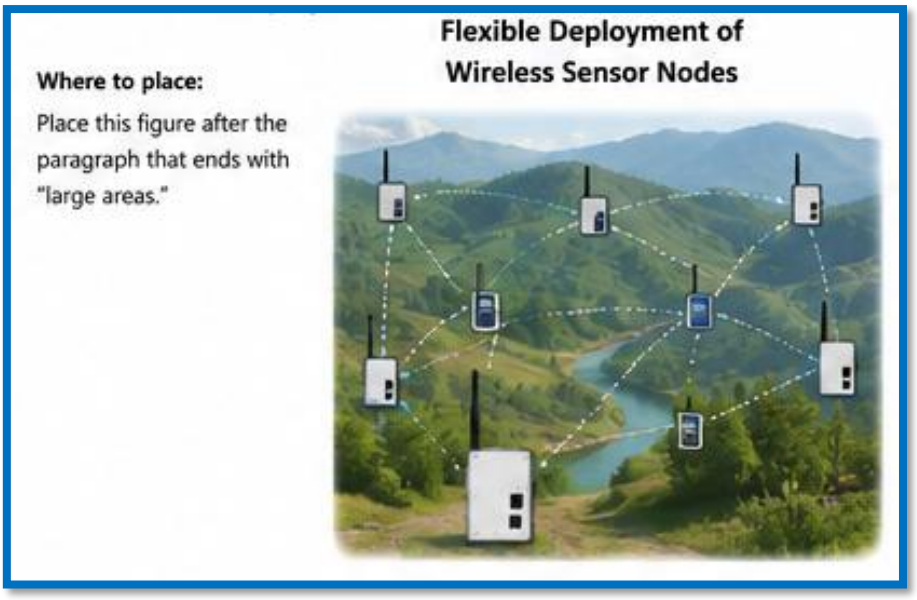

Where to place:
Place this figure after the paragraph that ends with "large areas."
Flexible Deployment of Wireless Sensor Nodes

Figure 2.22: Flexible Deployment of Wireless Sensor Nodes

2.13.4 Real-Time Monitoring

Wireless sensor networks enable continuous monitoring and rapid transmission of collected information.

Real-time monitoring improves situational awareness and supports timely decision-making processes.

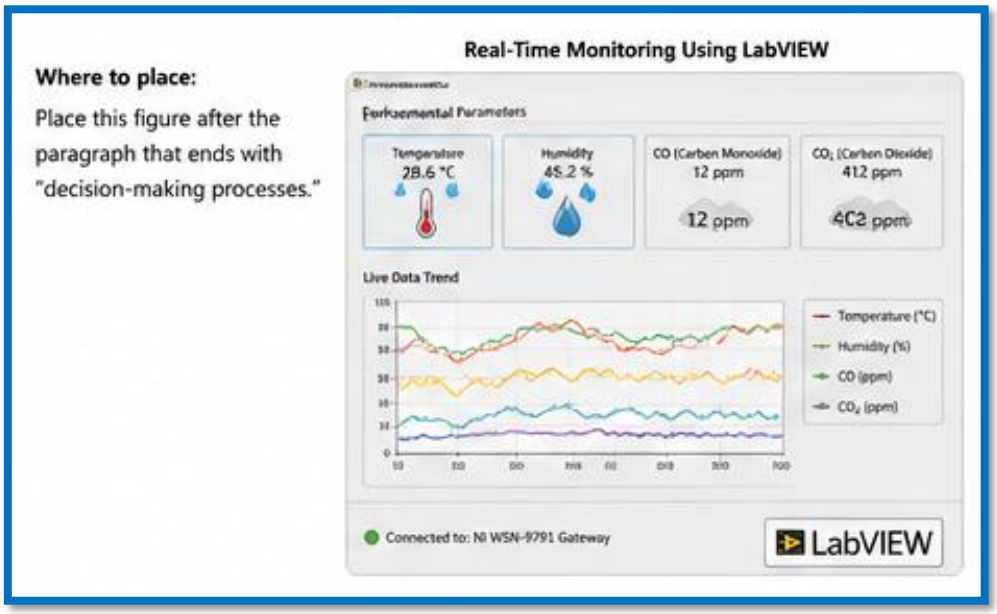


Figure 2.23: Real-Time Monitoring Using LabVIEW

2.13.5 Remote Accessibility

Data collected by wireless sensor networks can be accessed remotely through centralized monitoring systems.

Remote accessibility reduces the need for frequent field visits and improves operational efficiency.

2.13.6 Scalability and Expandability

Additional sensor nodes can be incorporated into the network as monitoring requirements evolve.

This capability enables gradual system expansion without major infrastructure modifications.

2.13.7 Energy Efficiency

Modern wireless sensor networks are designed to operate with minimal power consumption.

Low-power operation extends battery lifetime and supports long-term deployment in remote environments.

2.13.8 Reliability

Wireless sensor networks provide reliable communication mechanisms capable of operating under challenging environmental conditions.

Features such as error detection, retransmission mechanisms, and network management improve overall system dependability.

2.14 Communications Protocols

2.14.1 The Protocol Stack

Like any networked system, a WSN relies on a layered protocol stack. The layers — physical, data link (including MAC), network, transport, and application — are the same ones found in conventional networks, but the protocols at each layer have been substantially rethought to suit the constraints of sensor hardware. In addition to these vertical layers, cross-layer design has become an important paradigm in WSN research, allowing information to flow between layers in ways that would be considered bad practice in conventional networking but that can yield significant efficiency gains in resource-constrained environments .

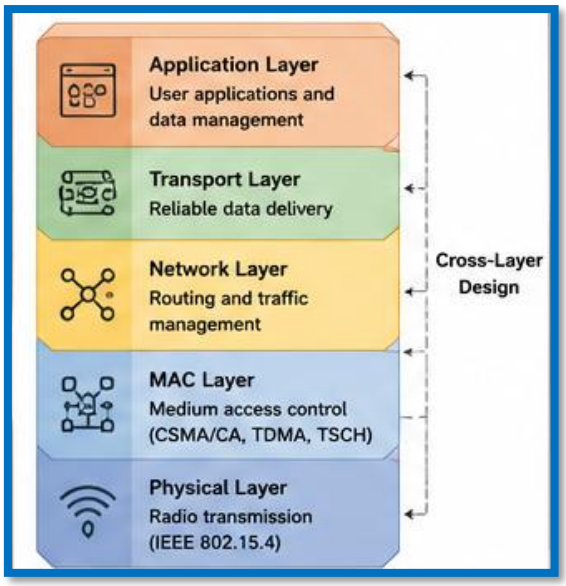


Figure 2.24: WSN Protocol Stack

2.14.2 Physical Layer

The physical layer handles the conversion of bits into radio signals and back again. For most WSN applications, the dominant standard is IEEE 802.15.4, which was designed specifically for low-rate wireless personal area networks. It operates in the 2.4 GHz band (as well as sub-GHz bands in some regions), uses Direct Sequence Spread Spectrum (DSSS) modulation, and supports data rates up to 250 kbps. These numbers are modest by modern networking standards, but they are more than adequate for the low-data-rate sensor applications that make up the bulk of WSN deployments.

Figure 2.25: IEEE 802.15.4 Physical Layer

One of the more interesting challenges at the physical layer is managing the trade-off between transmission power and communication range. Reducing transmission power saves energy, which is the dominant concern in most WSN designs, but it also reduces range, which may force the use of more hops to reach the sink and thereby increase end-to-end latency and energy consumption due to forwarding. Getting this balance right typically requires careful site-specific tuning .

2.14.3 MAC Protocols

The MAC layer controls access to the shared wireless medium. In WSNs, poor MAC design is one of the most common causes of excessive energy consumption. The main culprits are idle listening (keeping the radio on when there is nothing to receive), overhearing (receiving packets addressed to other nodes), and collisions (which require retransmission). A good WSN MAC protocol minimizes all three.

MAC protocols for WSNs fall into two broad families. Schedule-based protocols, such as those using Time Division Multiple Access (TDMA), assign each node a predetermined time slot for transmission. This eliminates collisions and allows nodes to sleep during slots assigned to others, which is excellent for energy efficiency. The downside is that TDMA requires synchronization across the network, which introduces its own overhead. IEEE 802.15.4e's Time-Slotted Channel Hopping (TSCH) mode is a well-known example of a schedule-based approach that also adds frequency diversity to improve robustness [3].

Contention-based protocols take the opposite approach: nodes compete for the channel using rules designed to minimize collisions. The standard IEEE 802.15.4 beacon-enabled mode uses slotted CSMA/CA for this purpose.

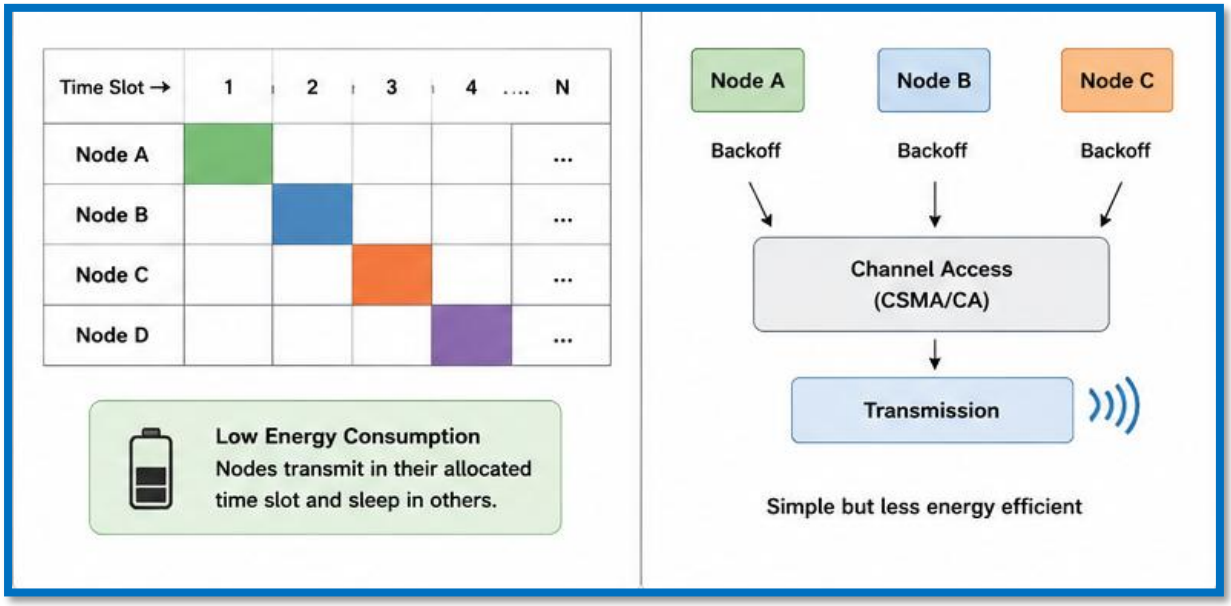


Figure 2.26: TDMA vs. CSMA/CA MAC Protocols

Contention-based protocols are simpler to implement and adapt more naturally to variable traffic loads, but they tend to be less energy-efficient than schedule-based protocols under light traffic conditions because nodes may still need to listen for extended periods.

Industrial WSN standards such as WirelessHART and ISA-100.11a are worth mentioning here. Both are built on the IEEE 802.15.4 physical layer but use TDMA scheduling with channel hopping at the MAC layer to achieve the high reliability required in process automation environments. They represent mature, commercially deployed examples of WSN protocol design.

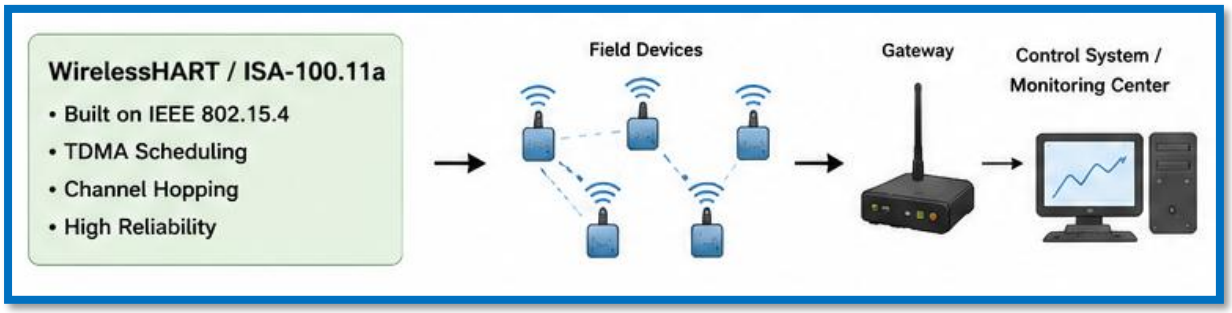


Figure 2.27: WirelessHART and ISA-100.11a Based on IEEE 802.15.4

2.14.4 Routing Protocols

Routing protocols in WSNs are generally classified into data-centric, hierarchical, and location-based protocols. These protocols are designed to provide reliable communication while minimizing energy consumption. In the proposed system, routing is handled internally by the NI WSN platform, allowing reliable wireless communication between sensor nodes and the gateway.

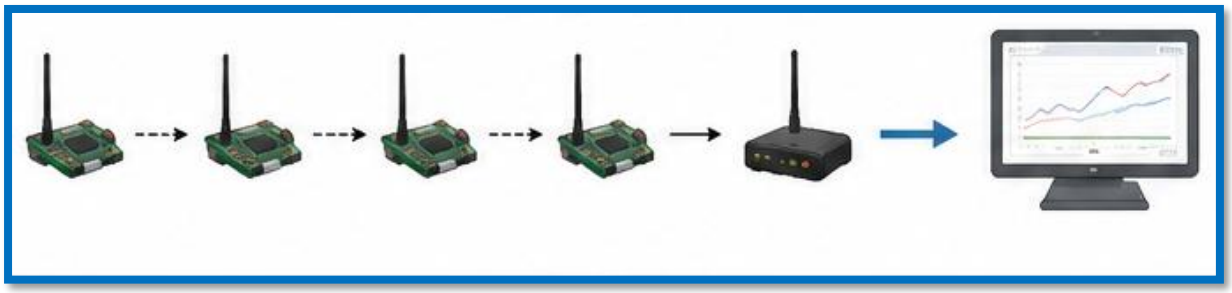


Figure 2.28: Routing Communication Between Sensor Nodes and the NI WSN Gateway

2.14.5 Transport and Application Layers

At the application layer, collected sensor data is transferred to LabVIEW for visualization, monitoring, and alarm generation.

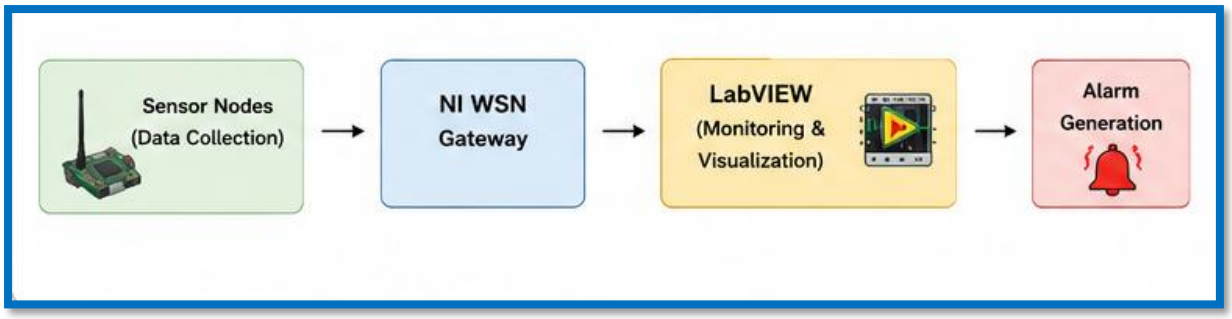


Figure 2.29: Data Flow from Sensor Nodes to LabVIEW

2.15 Infrastructure-less Wireless Sensor Networks

Wireless Sensor Networks are often deployed in remote environments where traditional communication infrastructure is unavailable. In such scenarios, sensor nodes organize themselves and communicate wirelessly to deliver environmental data to a gateway or sink node.

These infrastructure-less deployments are particularly suitable for applications such as forest monitoring, disaster management, and environmental surveillance, where installing conventional communication networks is impractical. The ability of WSNs to operate autonomously makes them an effective solution for early forest fire detection and continuous environmental monitoring.

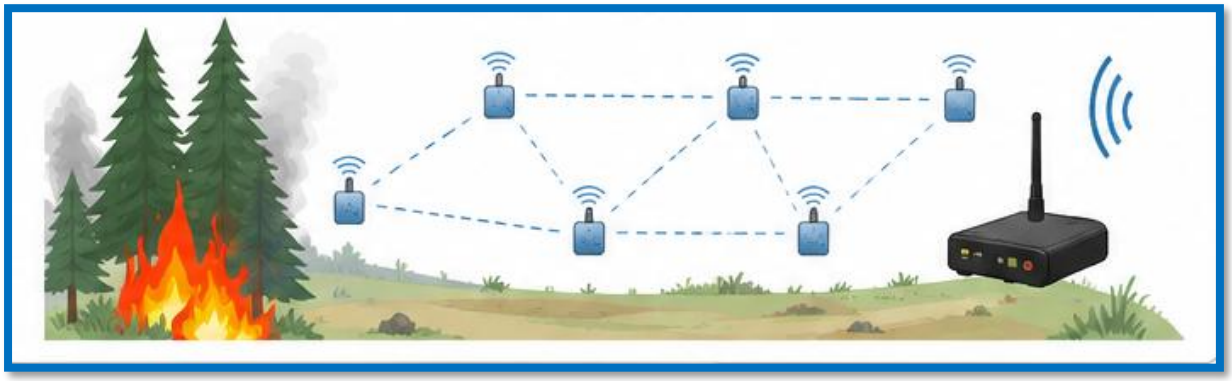


Figure 2.30: Infrastructure-less Wireless Sensor Network for Forest Monitoring

2.16 Chapter Summary

This chapter presented the fundamental concepts of Wireless Sensor Networks (WSNs) and their significance in modern environmental monitoring applications. It introduced the architecture, components, communication principles, and network topologies commonly used in WSNs, highlighting their advantages in terms of flexibility, scalability, low power consumption, and cost-effective deployment. The chapter also discussed the relationship between WSNs and Ad Hoc networks, emphasizing their shared characteristics and the role of self-organizing communication in infrastructure-independent environments.

Furthermore, the chapter described the hardware components used in the proposed system, including the NI WSN-3202 and NI WSN-3212 sensor nodes, along with the NI WSN-9791 gateway. Their operating principles, communication capabilities, and integration within the wireless network were explained. Finally, the chapter established the theoretical and technical foundation required for the implementation of the proposed early forest fire detection system, providing the necessary background for the software integration and system development presented in the following chapters.

CHAPTER

3

Introduction to LabVIEW

Graphical Programming • Virtual Instruments • Data Acquisition • Automation

Graphical Programming
Program by connecting icons and wires

Front Panel & Block Diagram
Design UI and build functionality visually

Debugging and Probes
Inspect data flow and troubleshoot with ease

subVI Creation & Usage
Create reusable and modular components

Loops & Case Structures
Control execution flow and make decisions

Graphs and File I/O
Visualize data and store it in multiple formats

Intuitive Environment | Powerful Analysis and Visualization | Hardware Integration | Real-World Applications

3.1 Introduction

LabVIEW (Laboratory Virtual Instrument Engineering Workbench) is a graphical system design and development environment developed by National Instruments (NI). Unlike conventional text-based programming languages such as C, C++, or Python, LabVIEW uses a graphical programming language called G, where applications are created by connecting functional nodes with wires in a Block Diagram based on the principles of dataflow programming. The resulting applications are known as Virtual Instruments (VIs), each consisting of a Front Panel, which serves as the user interface, and a Block Diagram, which contains the program logic.

LabVIEW has become one of the leading software platforms for measurement, automation, and control applications due to its intuitive graphical interface, extensive libraries for signal processing and data analysis, and seamless integration with a wide range of hardware devices. It is widely used in industrial automation, scientific research, data acquisition, instrument control, embedded systems, and real-time monitoring applications.

In Wireless Sensor Network (WSN) applications, LabVIEW functions as a central monitoring and control platform that acquires data from distributed sensor nodes, processes and analyzes the collected information, visualizes measurements through interactive dashboards, and stores data for future analysis. Its built-in communication protocols, hardware drivers, and visualization tools enable the rapid development of reliable monitoring systems without requiring extensive expertise in low-level programming, making it an ideal platform for implementing real-time environmental monitoring applications.

3.2 Overview of LabVIEW

3.2.1 History and Development

LabVIEW was introduced by National Instruments (NI) in 1986 for the Apple Macintosh platform. It revolutionized engineering software development by replacing conventional text-based programming with a graphical dataflow programming approach, allowing engineers to develop applications more intuitively through visual programming. Over the years, LabVIEW has undergone continuous development, expanding its compatibility to multiple operating systems, including Windows, Linux, and macOS. Its capabilities have also grown to support real-time operating systems, Field-Programmable Gate Arrays (FPGAs), embedded controllers, industrial communication protocols, and distributed data acquisition systems.

Today, LabVIEW is recognized as one of the most widely used engineering software platforms for measurement, automation, control, and testing applications. It is extensively employed in industrial automation, scientific research, biomedical engineering, telecommunications, aerospace, automotive systems, and academic laboratories due to its flexibility, reliability, and seamless integration with hardware devices.

3.2.2 LabVIEW Programming Model

Applications developed in LabVIEW are referred to as Virtual Instruments (VIs) because they replicate the functionality of traditional laboratory instruments while providing software-based flexibility. Every VI consists of two fundamental components:

- **Front Panel:** The graphical user interface (GUI) through which users interact with the application. It contains controls, such as buttons, switches, knobs, and numeric inputs, along with indicators including graphs, charts, LEDs, gauges, and numeric displays for presenting measurement results and system status.
- **Block Diagram:** The programming environment where the application logic is implemented using the **G** graphical programming language. It consists of interconnected function nodes, structures, constants, and wires that define the flow of data and the execution of operations.

Unlike sequential programming languages, LabVIEW follows the dataflow programming model, in which a function executes only after all required input data become available. Consequently, independent sections of a program can execute simultaneously, providing inherent parallelism without requiring explicit multithreading. This execution model significantly improves performance in applications involving real-time monitoring, high-speed data acquisition, multi-channel signal processing, and distributed control systems, making LabVIEW particularly suitable for Wireless Sensor Network (WSN) applications.

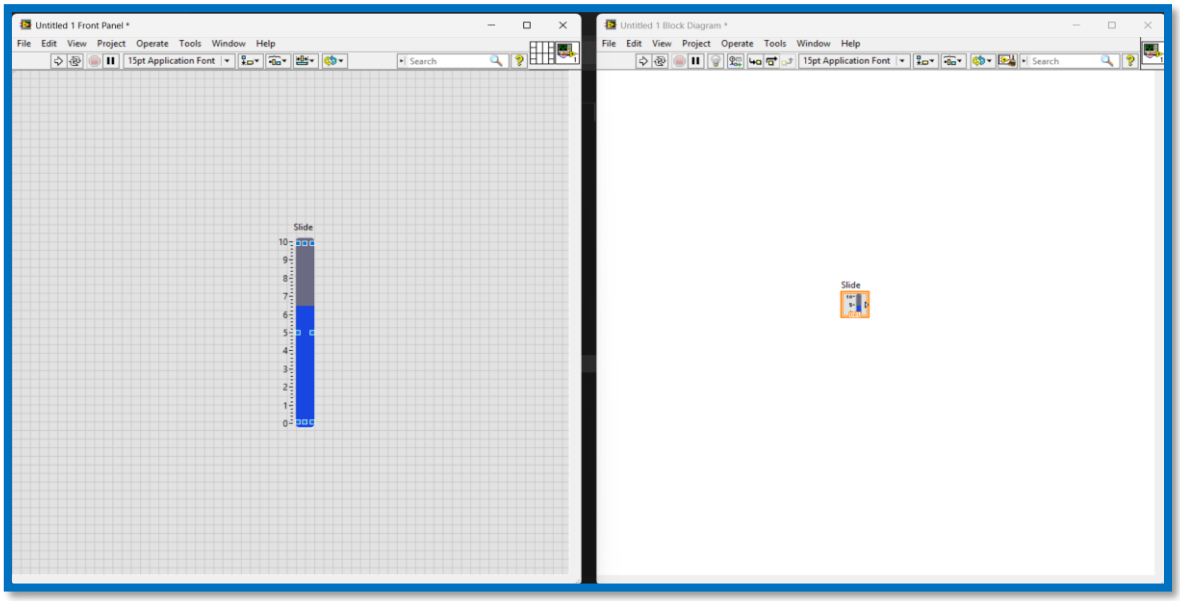


*Figure 3.1: **Front Panel (left) and Block Diagram (right) of a Virtual Instrument (VI).***

3.3 LabVIEW Key Components and Features

3.3.1 Virtual Instruments (VIs)

A Virtual Instrument (VI) is the fundamental programming unit in LabVIEW. Similar to functions or procedures in conventional programming languages, a VI encapsulates a specific task and can operate either as a standalone application or as a reusable SubVI within larger programs. This modular programming approach enhances code readability, promotes reusability, simplifies debugging, and improves the scalability of complex applications.

Each VI consists of four main elements:

- **Front Panel:** The graphical user interface (GUI) through which users interact with the application. It contains controls for data input and indicators for displaying results and system status.
- **Block Diagram:** The graphical source code where the program logic is implemented using interconnected functions, structures, and data wires.
- **Connector Pane:** Defines the input and output terminals of a VI, enabling it to be integrated as a SubVI in other applications.
- **Icon:** A graphical representation of the VI that appears on the block diagram when the VI is used as a SubVI.

3.3.2 Data Types and Wires

Communication between functions in LabVIEW is achieved through wires, which transfer data from one node to another according to the dataflow programming model. Each wire represents a specific data type, identified by its colour and appearance. Common data types include floating-point and integer numeric values, Boolean values, strings, arrays, and clusters.

LabVIEW automatically checks data compatibility during program development, preventing invalid connections between incompatible data types. This built-in type checking reduces programming errors and contributes to the reliability of measurement and control applications.

3.3.3 Programming Structures

LabVIEW provides a set of graphical programming structures that control the execution flow of applications while preserving the advantages of the dataflow paradigm. The most commonly used structures include:

- **While Loop:** Continuously executes the enclosed code until a specified stop condition is satisfied. It is widely used in continuous monitoring and data acquisition applications.

- **For Loop:** Repeats the enclosed code for a predetermined number of iterations, making it suitable for repetitive calculations and batch processing.
- **Case Structure:** Implements conditional execution based on the value of an input condition, providing functionality equivalent to *if–else* or *switch* statements in text-based programming languages.
- **Event Structure:** Executes code in response to user interface events, such as button presses or value changes, improving application responsiveness while minimizing unnecessary processor usage.

These structures enable developers to build flexible, efficient, and interactive engineering applications with minimal programming complexity.

3.3.4 Data Acquisition (DAQ) Subsystem

One of LabVIEW's most significant strengths is its seamless integration with Data Acquisition (DAQ) hardware through the NI-DAQmx driver. The DAQ subsystem provides a standardized interface for acquiring real-world signals, processing measurements, and controlling external devices.

The primary DAQ functionalities include:

- **Analog Input (AI):** Acquires analog signals from sensors such as temperature, humidity, pressure, and voltage sensors.
- **Analog Output (AO):** Generates analog control signals for actuators, motors, and other analog devices.
- **Digital Input/Output (DIO):** Reads and writes digital logic signals for switching, triggering, and industrial control applications.
- **Counter/Timer Operations:** Performs pulse counting, frequency measurement, event timing, and pulse generation for synchronization and timing applications.

These capabilities make LabVIEW an ideal platform for real-time measurement, industrial automation, and Wireless Sensor Network (WSN) applications, where reliable acquisition, processing, visualization, and storage of sensor data are essential.

3.4 LabVIEW in Wireless Sensor Networks

3.4.1 Role of LabVIEW in WSN Applications

Wireless Sensor Networks (WSNs) consist of multiple sensor nodes distributed over a monitored area to collect physical or environmental parameters such as temperature, humidity, pressure, gas concentration, and light intensity. These measurements are transmitted wirelessly to a central monitoring station, where they are processed, analyzed, and presented to the end user.

LabVIEW serves as an effective supervisory platform for WSN applications by providing a comprehensive environment for communication, data processing, visualization, and system control. Its graphical programming model, extensive hardware support, and built-in communication libraries enable the rapid development of reliable monitoring and automation systems.

The primary functions of LabVIEW in WSN applications include:

- **Data Acquisition:** Receiving measurement data from distributed wireless sensor nodes through various communication interfaces and hardware gateways.
- **Data Parsing and Processing:** Decoding incoming data packets, validating transmitted values, applying calibration factors when necessary, and converting raw sensor readings into meaningful engineering units.
- **Real-Time Visualization:** Presenting live sensor measurements using interactive graphical components such as waveform charts, graphs, gauges, numeric indicators, and status LEDs to provide continuous monitoring of the system.
- **Data Logging:** Recording acquired measurements in files or databases for historical analysis, performance evaluation, troubleshooting, and documentation.
- **Alarm and Event Management:** Continuously monitoring sensor values against predefined thresholds and generating visual or audible alarms whenever abnormal operating conditions are detected.
- **Control and Automation:** Sending control commands to remote sensor nodes or external actuators based on processed data, thereby supporting closed-loop monitoring and automated control applications.

Owing to these capabilities, LabVIEW has become one of the most widely adopted software platforms for developing real-time WSN monitoring systems in industrial, environmental, biomedical, and research applications.

3.4.2 Communication Interfaces

A major advantage of LabVIEW is its ability to communicate with a wide range of hardware devices and external applications through numerous communication protocols. This flexibility enables LabVIEW to integrate seamlessly with wireless sensor networks regardless of the underlying communication technology.

Serial Communication (UART / RS-232 / RS-485)

Serial communication remains one of the most widely used methods for connecting sensor nodes, embedded systems, and microcontrollers to a host computer. LabVIEW supports serial communication through the VISA (Virtual Instrument Software Architecture) library, which provides functions for configuring communication parameters such as baud rate, data bits, parity, stop bits, and flow control.

Using VISA, LabVIEW can transmit and receive serial data, making it suitable for interfacing with development boards, sensors, industrial controllers, and embedded systems that communicate through UART, RS-232, or RS-485 interfaces.

TCP/IP and UDP Communication

For distributed WSNs operating over Ethernet or wireless IP networks, LabVIEW provides built-in support for both Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) communication.

LabVIEW can operate as either a TCP/UDP client or server, allowing real-time data exchange with remote computers, embedded devices, cloud services, or other software applications.

- **TCP (Transmission Control Protocol):** Provides reliable, connection-oriented communication with guaranteed packet delivery and ordered data transmission. It is commonly used for applications requiring high reliability, such as remote monitoring, configuration, and control.
- **UDP (User Datagram Protocol):** Provides connectionless communication with minimal overhead, making it suitable for applications that require low latency and high-speed data streaming, where occasional packet loss is acceptable.

Wireless Communication Technologies

LabVIEW supports communication with a wide variety of wireless technologies commonly used in Wireless Sensor Networks. Depending on the application requirements, sensor nodes may communicate using protocols such as Wi-Fi, Bluetooth, Zigbee (IEEE 802.15.4), LoRa, or other proprietary wireless solutions.

In most cases, LabVIEW interfaces with these wireless networks through gateway devices or communication modules connected via USB, Ethernet, or serial interfaces. The received data are then decoded, processed, visualized, and stored using LabVIEW's built-in programming and data analysis capabilities.

The broad support for both wired and wireless communication technologies makes LabVIEW a versatile platform for developing scalable WSN applications across industrial automation, environmental monitoring, smart agriculture, healthcare, and Internet of Things (IoT) systems.

3.5 Data Acquisition and Processing in LabVIEW

3.5.1 Data Acquisition Pipeline

LabVIEW provides an integrated environment for acquiring, processing, visualizing, and storing data obtained from sensors and measurement devices. In a typical Wireless Sensor Network (WSN) application, the data acquisition process follows a structured sequence of operations that is repeatedly executed during system operation.

The general data acquisition pipeline consists of the following stages:

1. **System Initialization:** Communication interfaces, acquisition parameters, sampling rates, and data storage resources are configured before the monitoring process begins.
2. **Data Acquisition:** Sensor measurements are collected from remote sensor nodes through the selected communication interface, such as serial communication, Ethernet, TCP/IP, or wireless gateways.
3. **Data Parsing and Validation:** The received data packets are decoded into individual sensor values. Data integrity may be verified using validation techniques such as checksums or cyclic redundancy checks (CRC), depending on the communication protocol.
4. **Calibration and Unit Conversion:** Raw sensor readings are converted into standardized engineering units by applying calibration coefficients, scaling factors, or mathematical transformations.
5. **Real-Time Visualization:** The processed measurements are displayed on the Front Panel using graphical indicators, waveform charts, gauges, tables, and numeric displays to provide continuous system monitoring.
6. **Threshold Monitoring and Decision Making:** Measured values are continuously compared with predefined operating limits. When abnormal conditions are detected, LabVIEW can generate alarms, notify operators, or initiate automated control actions.

7. **Data Logging:** Time-stamped sensor data are stored in files or databases to facilitate historical analysis, reporting, and system diagnostics.
8. **Continuous Execution:** The acquisition cycle is repeatedly executed, typically within a While Loop, until the monitoring process is terminated by the user or the application.

This structured workflow enables LabVIEW to perform reliable real-time monitoring while ensuring efficient data management and system responsiveness.

3.5.2 Signal Processing

Beyond data acquisition, LabVIEW provides an extensive collection of built-in signal processing and analysis tools that enable engineers to extract meaningful information from raw sensor measurements. These functions can be applied in real time or during post-processing without requiring external software.

Common signal processing operations available in LabVIEW include:

- **Digital Filtering:** Removes unwanted noise and interference using filters such as low-pass, high-pass, band-pass, and notch filters.
- **Statistical Analysis:** Calculates descriptive statistics including mean, median, minimum, maximum, standard deviation, variance, and Root Mean Square (RMS) values for performance evaluation.
- **Frequency Analysis:** Performs spectral analysis using the Fast Fourier Transform (FFT) to convert time-domain signals into the frequency domain, facilitating the analysis of vibration, acoustic, and electrical signals.
- **Waveform Processing:** Supports waveform alignment, interpolation, resampling, synchronization, and comparison of multiple data channels.
- **Peak and Event Detection:** Identifies peaks, transients, anomalies, and other significant events that may indicate system faults or abnormal operating conditions.

These integrated processing capabilities enable LabVIEW to transform raw measurement data into meaningful information suitable for monitoring, diagnostics, and automated decision-making.

3.5.3 Data Logging and File I/O

Efficient data storage is an essential component of any measurement and monitoring system. LabVIEW provides comprehensive file input/output (File I/O) functions that enable acquired data to be recorded in various formats for documentation, analysis, and long-term storage.

Commonly supported data storage formats include:

- **TDMS (Technical Data Management Streaming):** A high-performance binary file format developed by National Instruments for storing large volumes of measurement data together with timestamps, channel information, and metadata.
- **CSV (Comma-Separated Values):** A widely supported text-based format that can be easily imported into applications such as Microsoft Excel, MATLAB, Python, and other data analysis tools.
- **Text Files:** Simple text-based files commonly used for storing event logs, alarm histories, configuration parameters, and system status information.
- **Database Connectivity:** LabVIEW supports direct communication with relational database systems, including MySQL, Microsoft SQL Server, PostgreSQL, and Oracle, through ODBC or the LabVIEW Database Connectivity Toolkit. This capability enables centralized storage, querying, and management of measurement data in large-scale monitoring applications.

The availability of multiple storage options allows developers to select the most appropriate data management strategy according to application requirements, data volume, and long-term analysis needs.

3.6 Introduction to Virtual Instruments and the LabVIEW Environment

LabVIEW programs are called Virtual Instruments, or VIs, because their appearance and operation imitate physical instruments, such as oscilloscopes and multimeters. LabVIEW contains a comprehensive set of tools for acquiring, analyzing, displaying, and storing data, as well as tools to help you troubleshoot your code.

When opening LabVIEW, you first come to the "Getting Started" window.

Figure 3.2: ***LabVIEW 2013 startup interface.***

In order to create a new VI, select "Blank VI" or in order to create a new LabVIEW project, select "Empty project". When you open a blank VI, an untitled front panel window appears. This window displays the front panel and is one of the two LabVIEW windows you use to build a VI. The other window contains the block diagram. The sections below describe the front panel and the block diagram.

How VIs Function

Dataflow Programming: In LabVIEW, nodes on the block diagram execute only when all required input data become available. Because multiple nodes can have their data inputs satisfied simultaneously, the language is inherently capable of parallel execution.

Modularity: Because each VI can be tested independently and has defined inputs and outputs via its connector pane, VIs are highly modular and can be easily embedded as subroutines (SubVIs) into larger, more complex programs.

Key Interface Elements

Controls Palette: Only available on the Front Panel; used to place inputs and outputs.

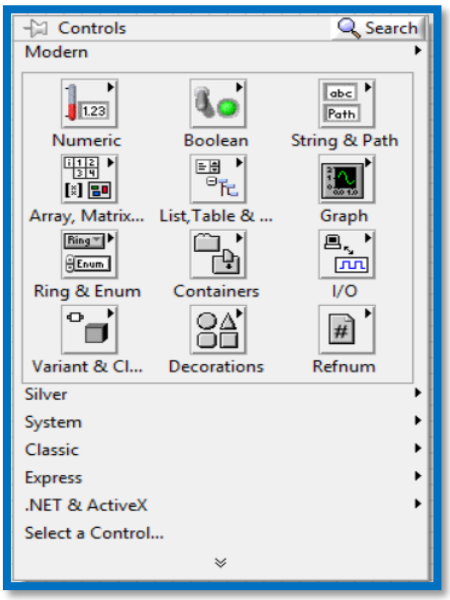


*Figure 3.3: **Controls Palette: Only available on the Front Panel.***

Functions Palette: Only available on the Block Diagram; used to place operations, functions, and structures.

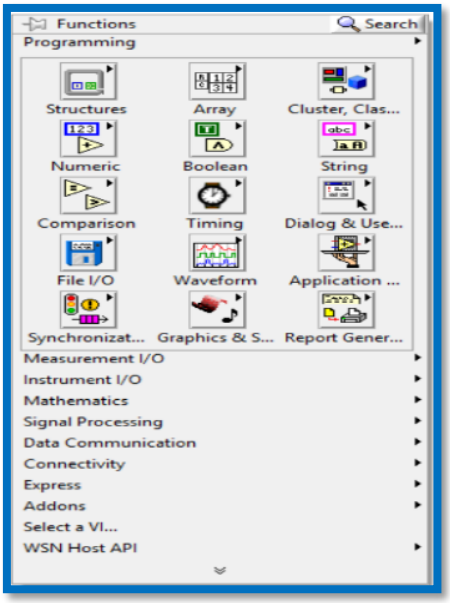


*Figure 3.4: **Functions Palette: Only available on the Block Diagram.***

You may Pin the palette, so it is always visible, just click the little pin button in the upper left corner of the palette.

Tools Palette: A floating window to switch between interaction modes (Operating, Positioning, Labeling, Wiring).

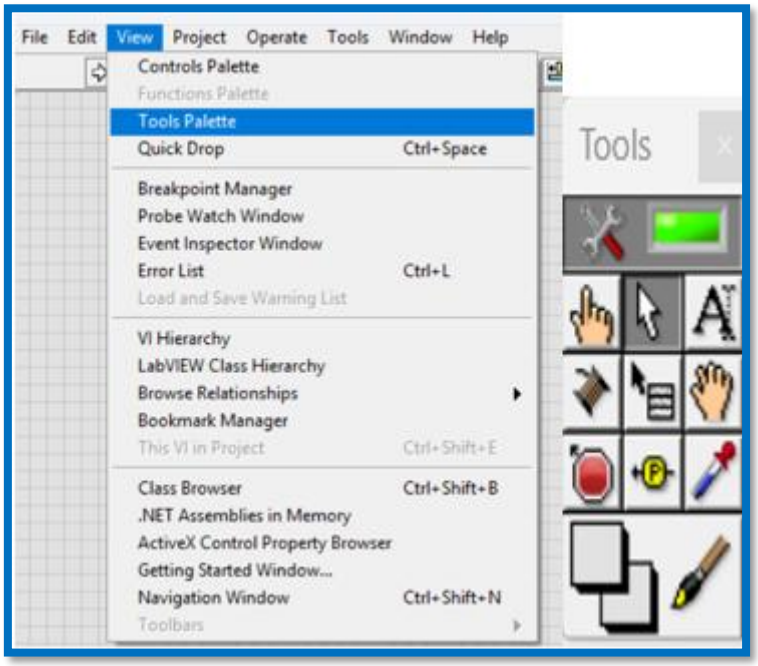


Figure 3.5: ***The Tools Palette.***

Toolbar & Execution Controls

The toolbar at the top of the window provides the primary interaction controls for managing program execution.

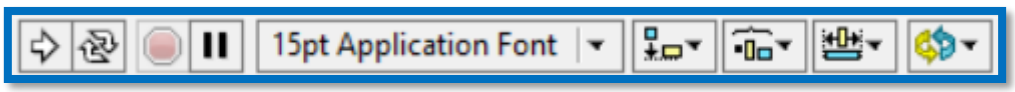


Figure 3.6: ***The toolbar and execution controls.***

3.7 Building Front Panel and Block Diagram

3.7.1 The Front Panel

Steps to Build:

1. Open the Controls Palette: Right-click anywhere on the blank Front Panel workspace.
2. Select Objects: Navigate through the categories (Numeric, Boolean, String, etc.) to select your desired components.
3. Placement: Click and drag the chosen control (input, e.g., knob, switch) or indicator (output, e.g., graph, LED) onto the panel.
4. Customization: Use the Labeling tool to rename objects and the Positioning tool to resize or rearrange them for a professional layout.

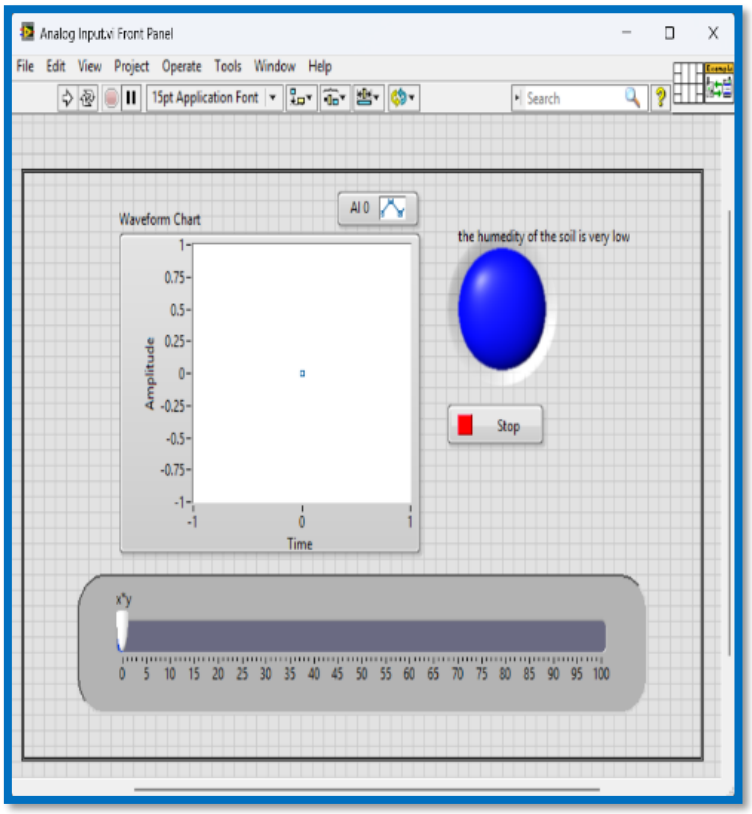


Figure 3.7: ***Building the Front Panel.***

3.7.2 The Block Diagram

Steps to Build:

1. Switch Windows: Press Ctrl + E to toggle between the Front Panel and the Block Diagram.
2. Locate Objects: When you place an object on the Front Panel, a corresponding terminal icon automatically appears on the Block Diagram

3. Open the Functions Palette: Right-click on the blank Block Diagram workspace to access mathematical, logical, and structural functions (e.g., While Loops, Add, Multiply).
4. Wiring Logic: Use the Wiring tool to connect the terminals of your inputs to the inputs of your functions, and the outputs of your functions to the terminals of your indicators.
5. Dataflow: Remember that code execution follows the path of the wires. A function executes only when all its required input data is received.

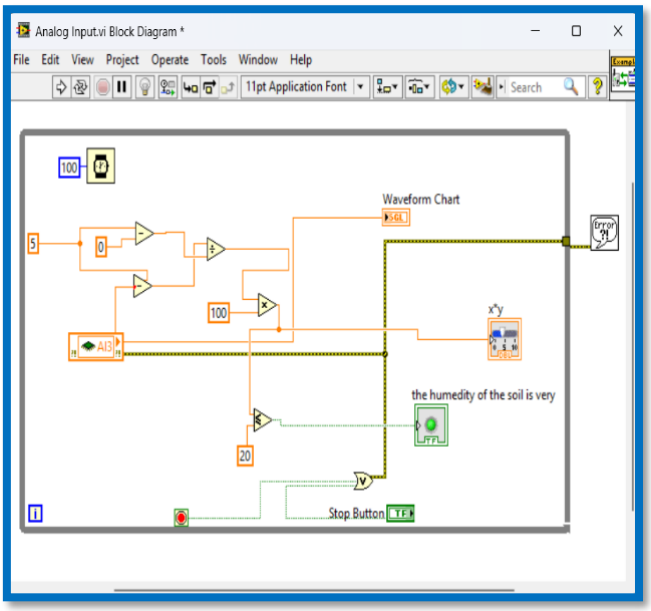


Figure 3.8: ***The Block Diagram.***

3.8 Running and Debugging VIs

3.8.1 Running a VI

Once your Front Panel and Block Diagram are complete, use the toolbar to execute the program:

Run Button (White Arrow): Click the Run button to execute the VI. If the Run arrow appears broken, it indicates that the VI contains one or more errors and cannot be executed until all errors have been corrected.

Figure 3.9: ***Run Button and Broken Run Arrow.***

Run Continuously: Useful for testing loops or real-time data acquisition; the VI will restart immediately after finishing.

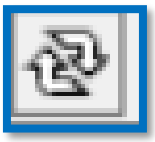


Figure 3.10: ***Run Continuously.***

Abort Execution: Use this to stop a VI immediately.

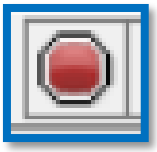


Figure 3.11: ***Abort Execution.***

To pause a running VI, click the Pause button. When paused, LabVIEW will highlight the current execution location on the block diagram and the button will turn red. To resume execution, click the button again.

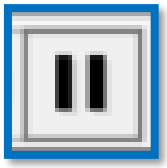


Figure 3.12: ***Pause Button.***

Note: It is best practice to include a stop button in your code rather than relying on this button.

3.8.2 Debugging Techniques

When a VI does not behave as expected, LabVIEW provides several powerful tools to inspect code execution:

Primary Debugging Tools:

Execution Highlighting: Click the Highlight Execution button (resembles a light bulb) on the block diagram. This slows down execution and animates the data flow, showing how values move from node to node.

Figure 3.13: ***Highlight Execution Button.***

Probes: Right-click any wire and select Probe. This opens a window that displays the real-time value passing through that specific wire during execution.

Breakpoints: Select the Breakpoint tool from the Tools palette and click on a wire or node. The VI will pause execution at that point, allowing you to inspect data and step through the code.

Single-Stepping: When paused at a breakpoint, use the stepping buttons on the toolbar to execute the block diagram one node at a time.

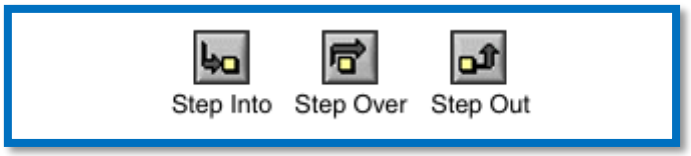


Figure 3.14: ***Debugging Tools.***

Troubleshooting Broken VIs

If the Run button appears broken, click it to open the Error List window. This list details every error in the code (e.g., unconnected wires, missing inputs), and clicking an error typically highlights the problematic area on the block diagram.

3.9 Creating VIs and SubVIs

3.9.1 Creating a Virtual Instrument (VI)

Every LabVIEW program is a VI. To create one, start by launching LabVIEW and selecting File → New VI.

Front Panel: Design the user interface by placing controls (inputs) and indicators (outputs) from the Controls Palette.

Block Diagram: Define functionality by wiring functions and structures from the Functions Palette

3.9.2 Creating and Using SubVIs

A SubVI is a VI that is called by another VI. Using SubVIs is essential for modularizing code, improving readability, and reusing logic.

Step-by-Step: Converting a VI to a SubVI

1.Design the Logic: Build a functional VI that performs a specific task.

2.Configure the Connector Pane:

- Right-click the icon in the top right corner of the Front Panel.
- Select Show Connector.
- The pattern determines how many inputs and outputs the SubVI will have.
- Click a terminal on the connector pane, then click the control or indicator on the Front Panel you want to assign to that terminal.

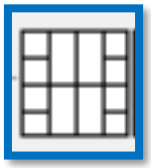


Figure 3.15: ***Connector Pane.***

3.Assign an Icon: Double-click the icon to open the Icon Editor. Create a descriptive graphical representation so you can easily identify the function when it is used later.

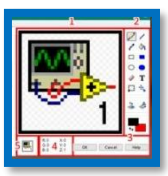


Figure 3.16: ***VI Icon (as it appears on the Block Diagram).***

4.Use as SubVI: To use your new VI in another program, simply drag and drop the VI file from your project explorer or file folder onto the Block Diagram of the main VI.

Figure 3.17: ***SubVI (used in the Block Diagram).***

3.10 Structures and Loops

Structures are nodes that control the execution flow of your code. They are essential for repeating tasks, making decisions, or managing data execution timing.

The figure below illustrates the main types of structures and loops available in LabVIEW.

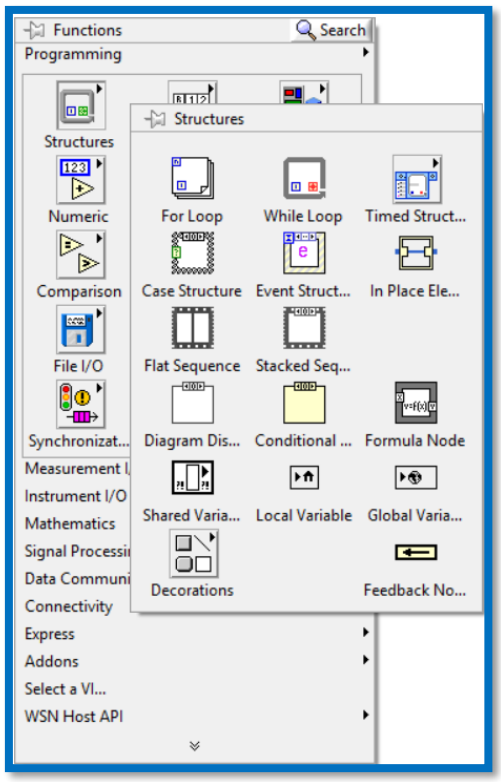


Figure 3.18: ***Structures and Loops.***

- For Loop: Used to execute a block of code a specific number of times. It is ideal for iterating over arrays or collections.

- While Loop: Executes a block of code repeatedly until a specific condition (e.g., a Stop button) is met.

- Timed Structure: Allows precise timing for loop execution, offering control over execution priority and timing constraints.

- Case Structure: Functions like an "if-else" or "switch" statement, executing one of several sub diagrams based on an input value or condition.

- Event Structure: Waits for specific events (like a button click or value change) and executes the corresponding code block.

- In Place Element: Used to modify data in memory without creating copies, which optimizes performance and reduces memory usage.

- Flat Sequence: Executes frames from left to right, one after another, ensuring a specific order of operations in the data flow.

- Stacked Sequence: Executes frames sequentially (0, 1, 2, etc.), but only one frame is visible at a time. Less common in modern code.

- Diagram Disable: Allows you to comment out or bypass portions of the block diagram during execution without deleting them.

- Conditional Disable: Allows parts of the code to be enabled or disabled based on platform, version, or user-defined symbols.

- Formula Node: Allows you to enter mathematical expressions directly using C-like syntax, simplifying complex math operations.

- Shared Variable: Used to share data between different VIs or parallel loops that are not directly wired together.

- Local Variable: Allows you to read or write to a front panel control or indicator from multiple locations within the same VI.

- Global Variable: Allows you to share data among multiple different VIs running simultaneously in memory.

- Decorations: Used for visual organization on the diagram (like lines, boxes, or labels) to improve readability.

- Feedback Node: Used to pass data from the current iteration of a loop to the next iteration, acting as a memory element.

3.11 Grouping Data

3.11.1 Grouping Data with Strings

- Strings on the Front Panel: User interface elements for text display and input.
- String Display Types: Different modes to visualize characters (e.g., normal, password, codes).
- Tables: A structure to organize multiple strings in a grid/matrix format.
- Editing, Formatting, and Parsing: Techniques to manipulate, convert, and extract data from strings.

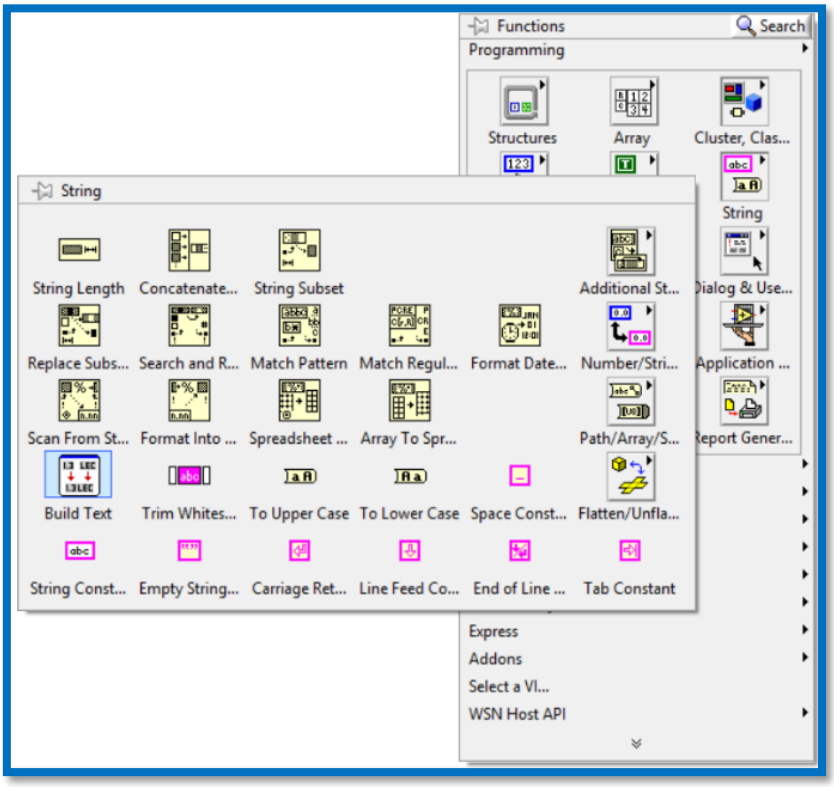


Figure 3.19: ***Grouping Data with Strings.***

3.11.2 Grouping Data with Arrays

- Restrictions: Rules on data types (must be the same type).
- Indexes: Used to access specific elements within an array.
- Examples of Arrays: Practical demonstrations of array usage.
- Creating Array Components: Defining controls, indicators, and constants for arrays.
- Multidimensional Arrays: Storing data in more than one dimension (e.g., matrices).
- Array Functions: Built-in operations to process or modify array data.
- Default Data: The initial values set within an array

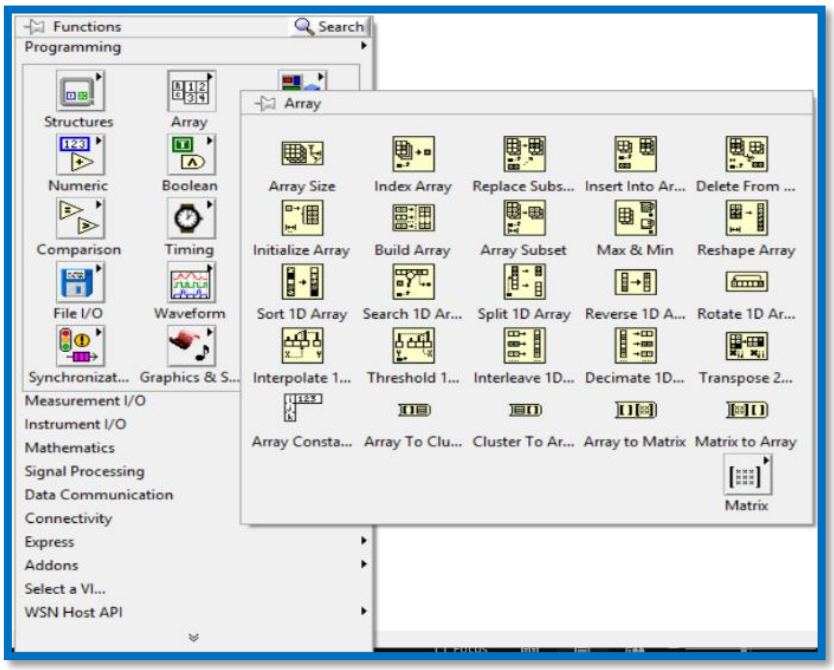


Figure 3.20: ***Grouping Data with Arrays.***

3.11.3 Grouping Data with Clusters

- Order of Cluster Elements: The sequential arrangement of data within a cluster.
- Cluster Functions: Operations to bundle or unbundle different data types.
- Creating Cluster Components: Defining clusters for UI controls, indicators, or constants.

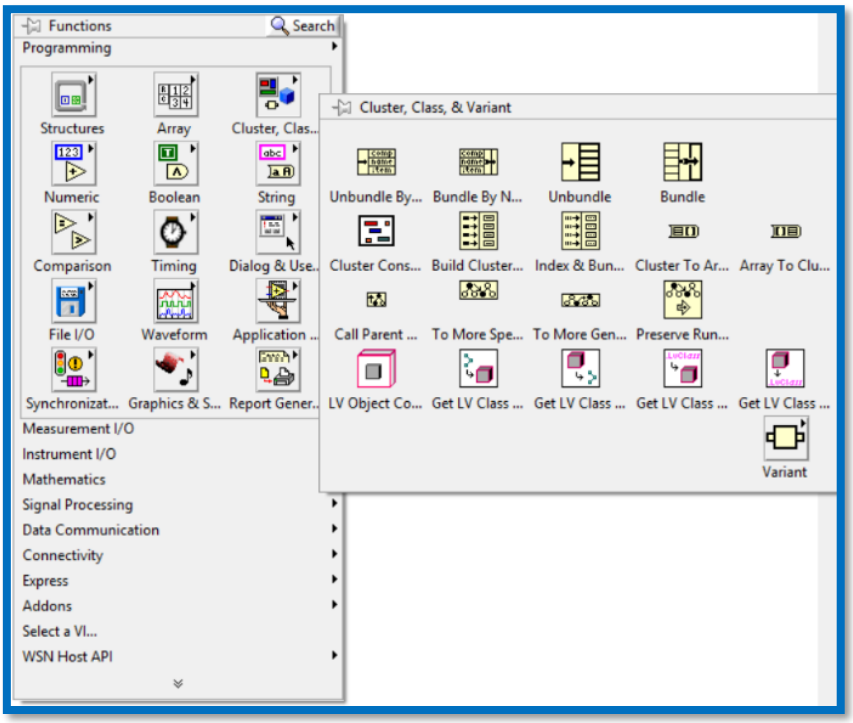


Figure 3.21: Grouping Data with Clusters.

3.12 Graphs and Charts

3.12.1 Types of Graphs and Charts

- Waveform Graphs/Charts: Tools for plotting data over time.
- Waveform Data Type: Special data structure for handling timing information.
- XY Graphs: Plotting arbitrary data points against each other.
- Intensity Graphs/Charts: Visualizing 3D data in 2D using color intensities.
- Digital Waveform Graphs: Displaying logic states (0s and 1s) over time.
- 3D Graphs: Visualizing data in a three-dimensional space.

3.12.2 Customizing Graphs and Charts

- Scaling & Formatting: Using multiple scales, autoscaling, and custom formatting for axes.
- Graph Palette: Built-in UI tools for interacting with graphs.
- Appearance: Customizing visual styles and layouts.
- Advanced Features:
- Graphs: Using cursors and annotations for precise measurement.
- Charts: Configuring history length, update modes, and plotting styles (overlaid vs. stacked).

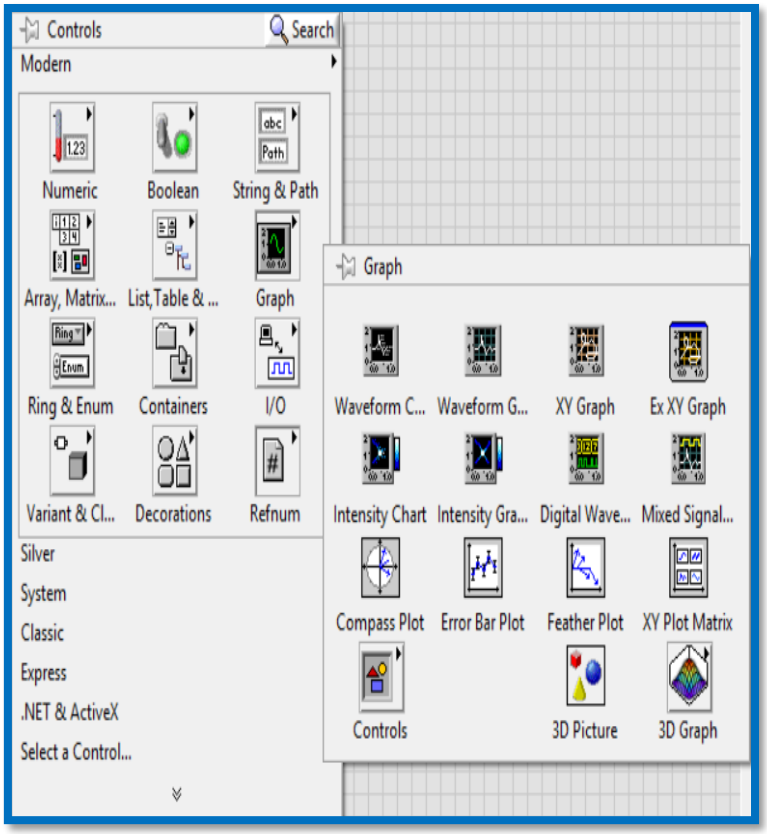


Figure 3.22: ***Graphs and Charts.***

3.13 File I/O

3.13.1 Core Concepts

- Basics of File I/O: Understanding how LabVIEW interacts with the file system.
- Choosing a File I/O Format: Selecting between text, spreadsheet, binary, or datalog formats based on your needs.
- VIs and Functions: Using built-in LabVIEW VIs for standard file operations (open, read, write, close).
- Storage VIs: Dedicated tools for managing file storage tasks.

3.13.2 File Types and Data Handling

- Text and Spreadsheet Files: Methods for formatting, writing, and scanning human-readable data (e.g., CSV).
- Binary Files: Storing data in a machine-readable format for efficiency.
- Datalog Files: Specialized files for storing complex, record-oriented data.
- Waveform File I/O: Specific workflows for writing and reading waveform data (timing and signal data).

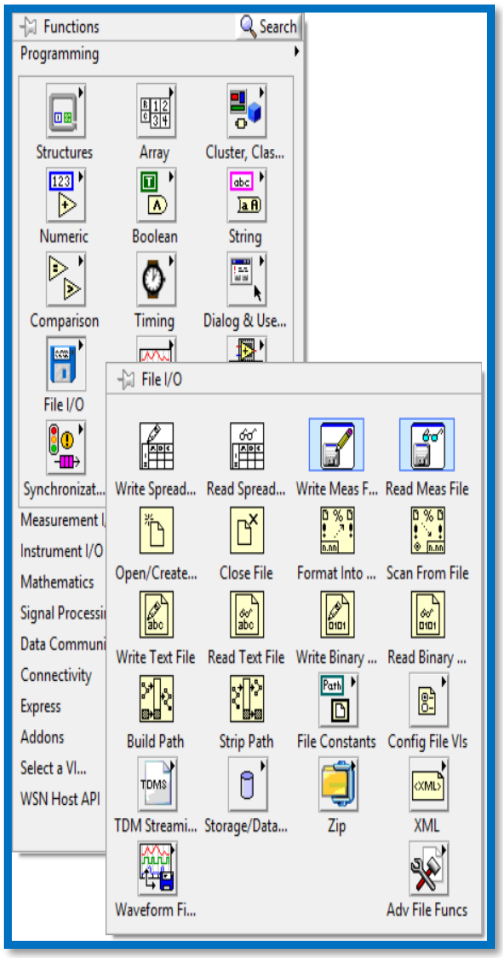


Figure 3.23: ***File I/O***

3.14 Documenting and Printing

3.14.1 To add a description to a VI, open the VI Properties dialog and navigate to the Documentation category.

Steps:

1. Open your VI in LabVIEW.
2. Go to File → VI Properties.

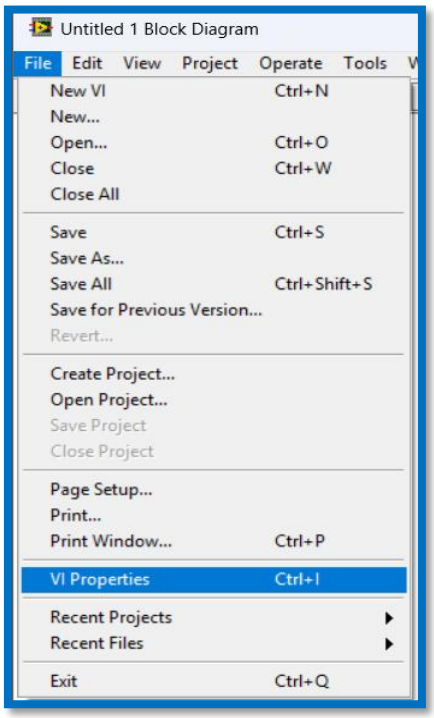


Figure 3.24: File Menu Showing VI Properties Option.

3. Select the Documentation category from the left panel.

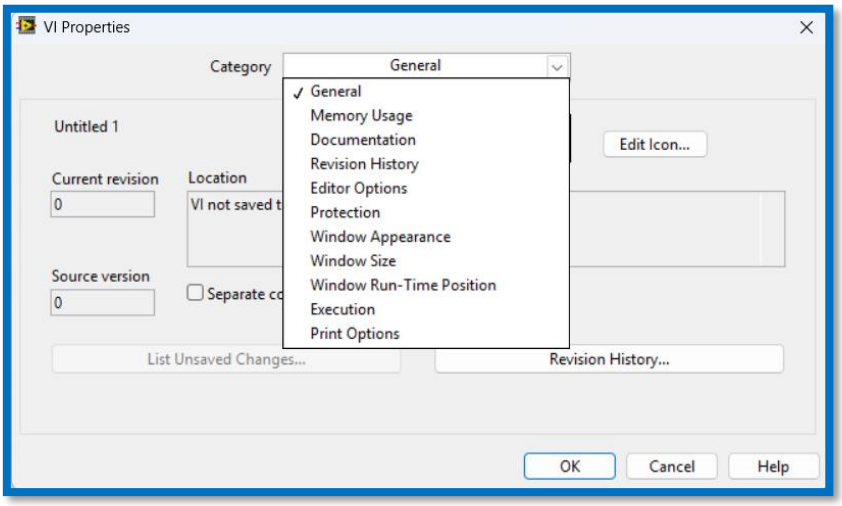


Figure 3.25: VI Properties Dialog – Category Selection

4. Enter a description in the “VI description” field.

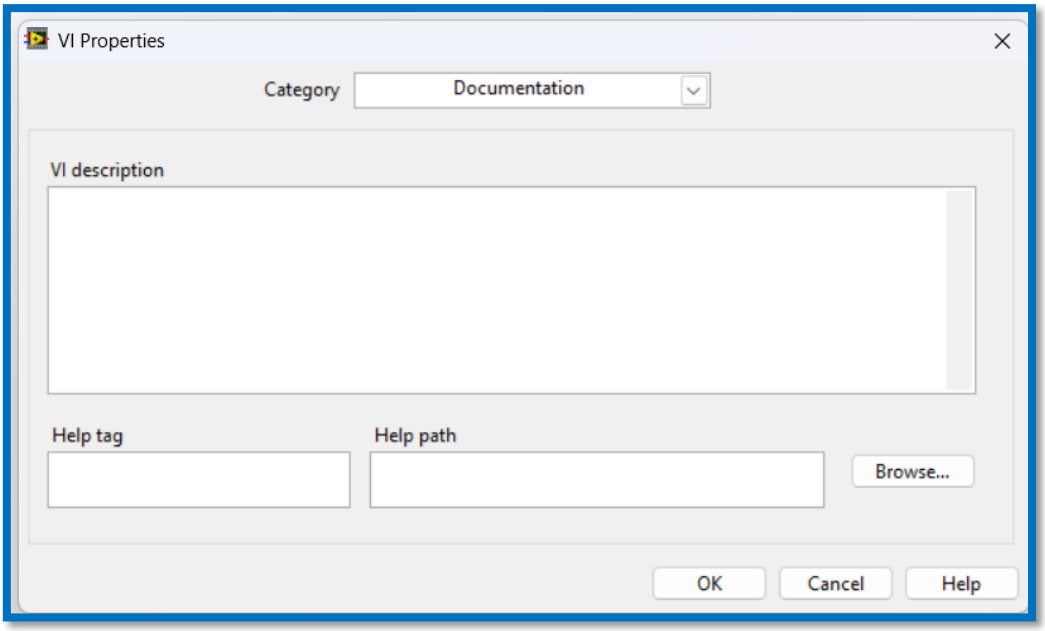


*Figure 3.26: **VI Properties Dialog – Documentation Category.***

5. Click OK to save.

3.14.2 Printing VIs: LabVIEW allows you to print the Front Panel, Block Diagram, and documentation from the File → Print dialog.

Steps:

1. Open your VI in LabVIEW.
2. Go to File → Print.

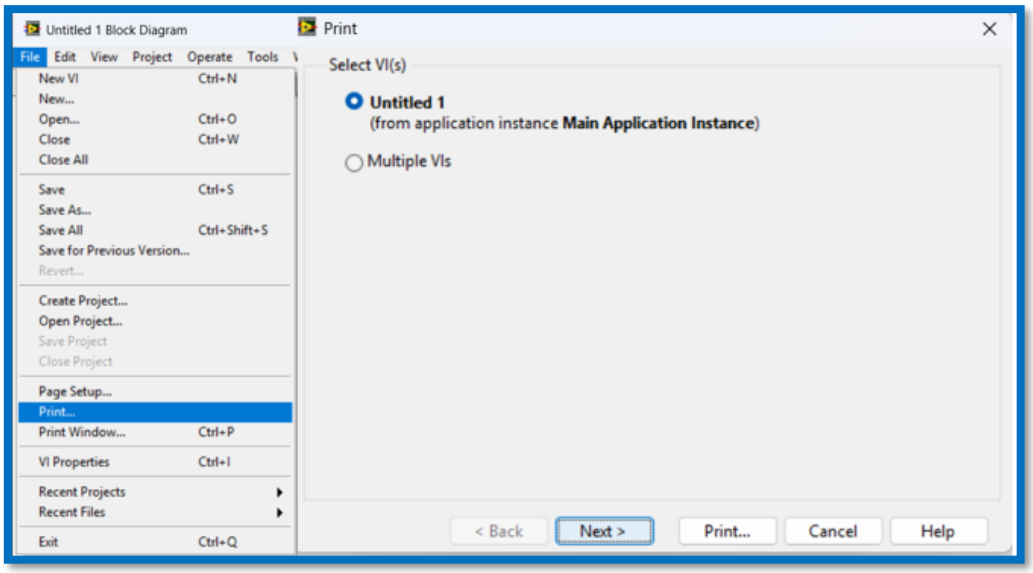


*Figure 3.27: **File Menu Showing Print Option.***

3. Choose what to print: Front Panel, Block Diagram, or both.

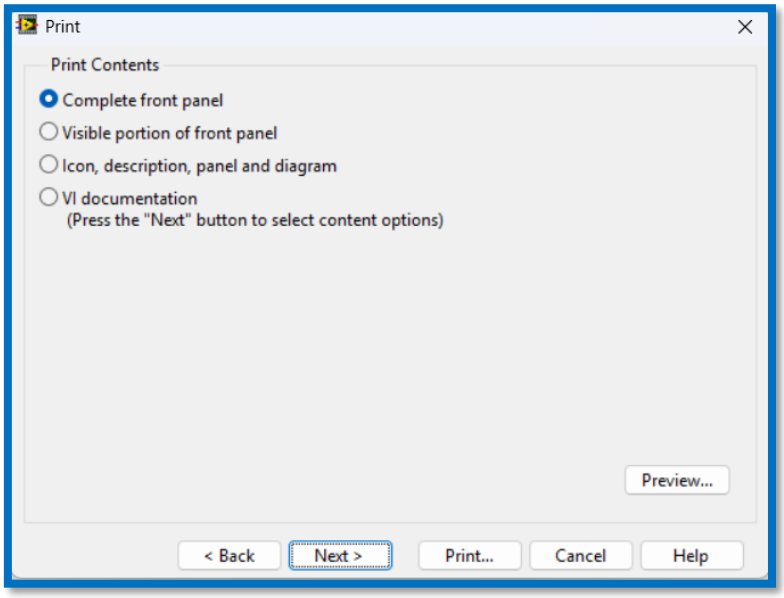


Figure 3.28: ***Print Dialog – Select VI and Print Contents.***

4. Select the printer or save as PDF.

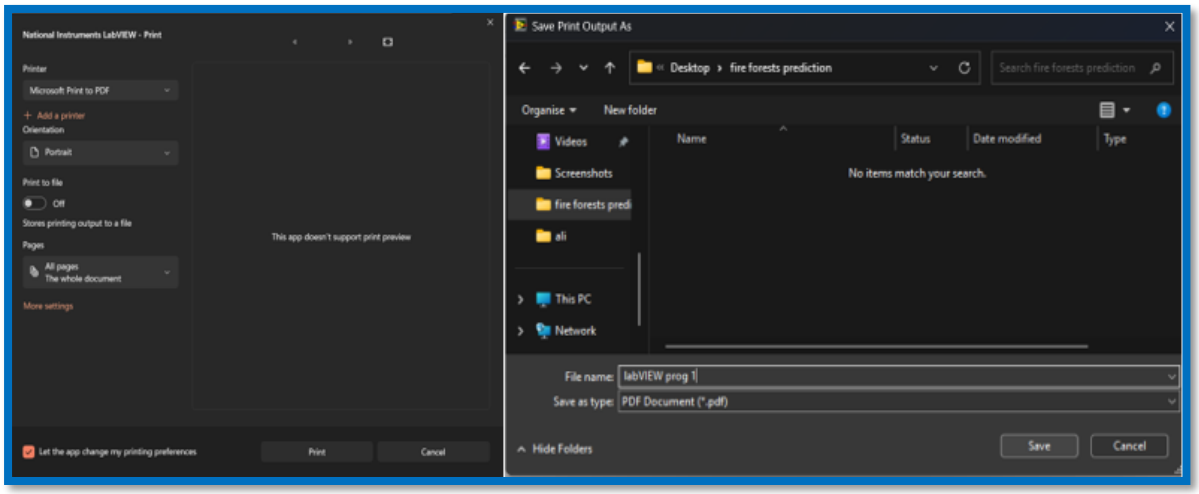


Figure 3.29: ***Saving the Print Output as PDF.***

5. Click Print.

3.15 Connecting the NI WSN-9791 Gateway to LabVIEW

4.15.1 Prerequisites

Before establishing communication between LabVIEW and the NI WSN-9791 gateway, the required hardware and software components must be prepared.

Hardware

- NI WSN-9791 Gateway
- NI WSN sensor nodes
- 9–30 V DC power supply
- Ethernet cable

Software

- NI-WSN Driver (Version 1.1 or later)
- LabVIEW (Version 8.6.1 or later)
- NI Measurement & Automation Explorer (NI MAX)

NI MAX is used to verify that the gateway is detected correctly by the host computer before integrating it into a LabVIEW project.

4.15.2 Gateway Configuration

The gateway must first be configured and verified through Windows network settings and NI MAX before it can be accessed from LabVIEW.

Step 1: Configure the Ethernet Connection

Connect the NI WSN-9791 gateway to the host computer using an Ethernet cable. Configure the IPv4 settings of the Ethernet adapter so that it belongs to the same subnet as the gateway.

Open control panel → Network and sharing center → Change adapter settings

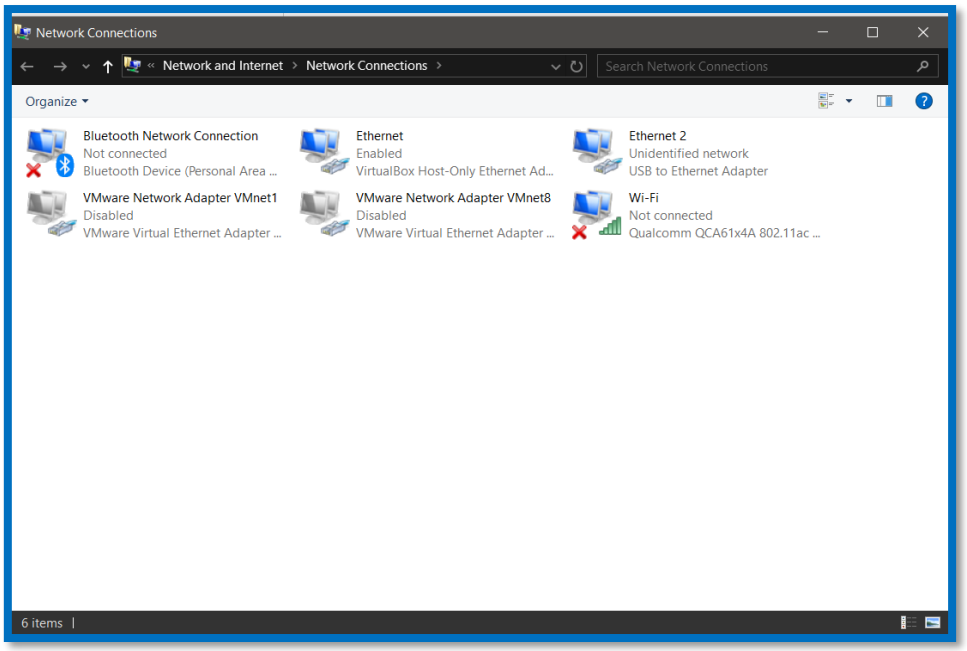


Figure 3.30: Network Connection Settings

On Ethernet card open properties → internet protocol version 4 (TCP/IPv4)

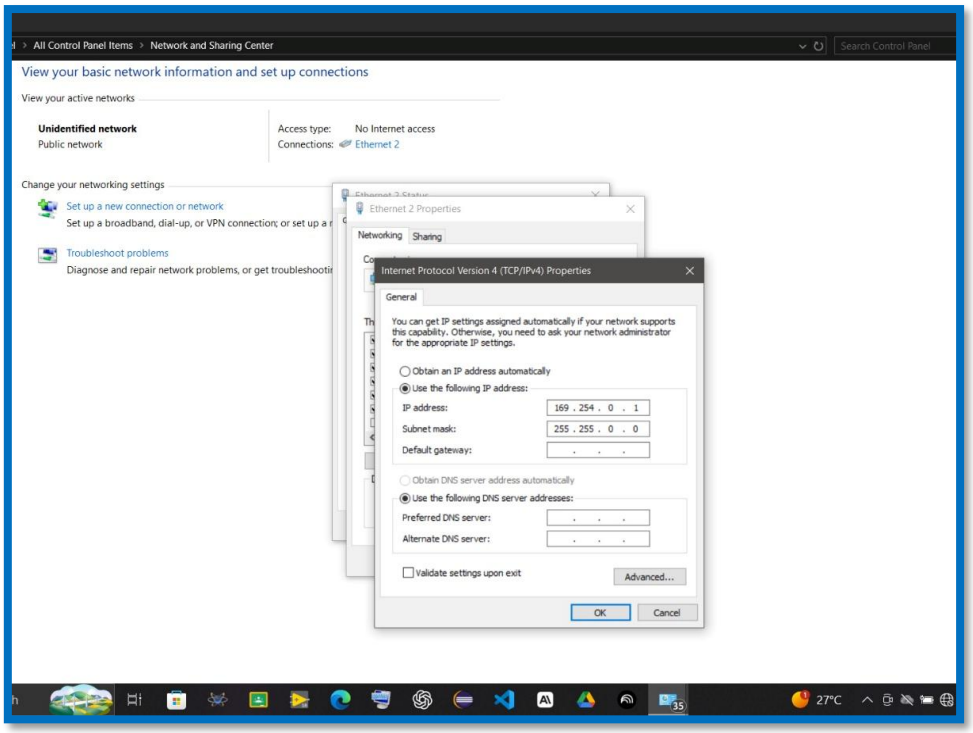

All Control Panel Items › Network and Sharing Center
Search Control Panel
View your basic network information and set up connections
View your active networks
Unidentified network
Public network
Access type: No Internet access
Connections: Ethernet 2
Change your networking settings
Set up a new connection or network
Set up a broadband, dial-up, or VPN connection; or set up a r
Troubleshoot problems
Diagnose and repair network problems, or get troubleshootir
Ethernet 2 Properties
Networking
Sharing
Internet Protocol Version 4 (TCP/IPv4) Properties
General
You can get IP settings assigned automatically if your network supports this capability. Otherwise, you need to ask your network administrator for the appropriate IP settings.
Obtain an IP address automatically
Use the following IP address:
IP address: 169 . 254 . 0 . 1
Subnet mask: 255 . 255 . 0 . 0
Default gateway:
Obtain DNS server address automatically
Use the following DNS server addresses:
Preferred DNS server:
Alternate DNS server:
Validate settings upon exit
Advanced...
OK
Cancel
27°C

Figure 3.31: IPv4 Configuration for Communication with the NI WSN-9791 Gateway

Step 2: Verify the Host System

Launch NI Measurement & Automation Explorer (NI MAX) and verify that the host computer is correctly recognized and ready to communicate with National Instruments devices.

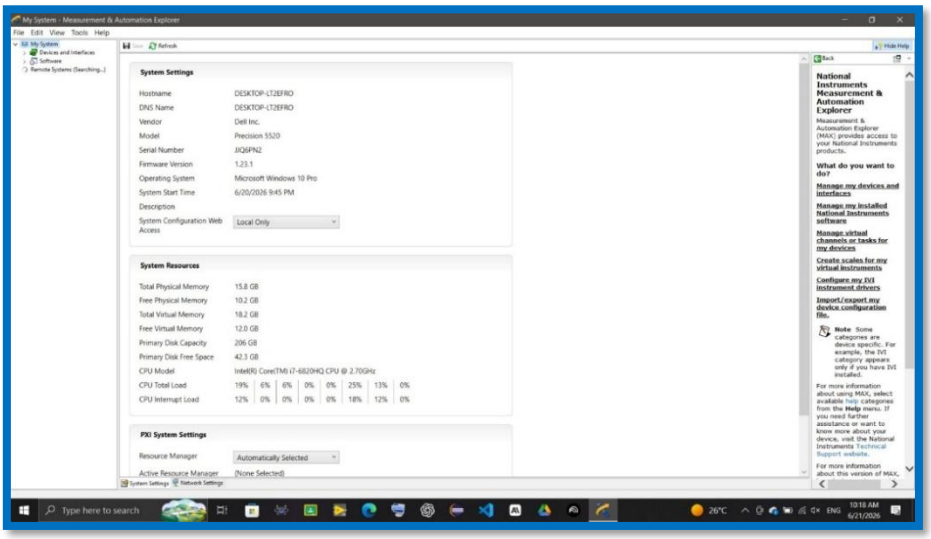


Figure 3.32: NI MAX System Settings for the Host Computer.

Step 3: Detect the NI WSN-9791 Gateway

Open the **Network Devices** section in NI MAX and verify that the NI WSN-9791 gateway is detected successfully. The gateway status should appear as **Connected – Running**, indicating successful communication with the host computer.

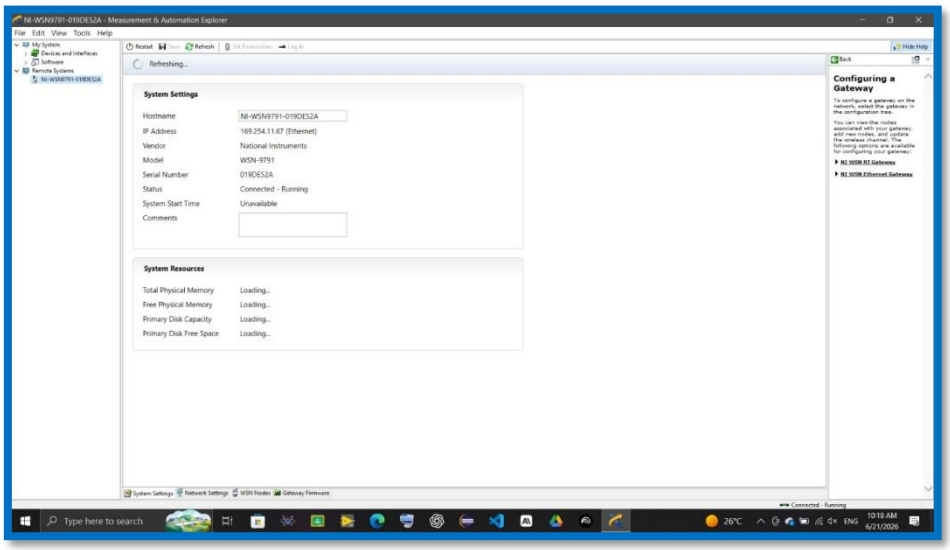


Figure 3.33: NI WSN-9791 Gateway System Settings in NI MAX

Step 4: Configure Wireless Sensor Nodes

Navigate to the **WSN Nodes** tab to detect and configure the wireless sensor nodes associated with the gateway. This interface allows sensor nodes to be added, monitored, and managed before starting data acquisition in LabVIEW.

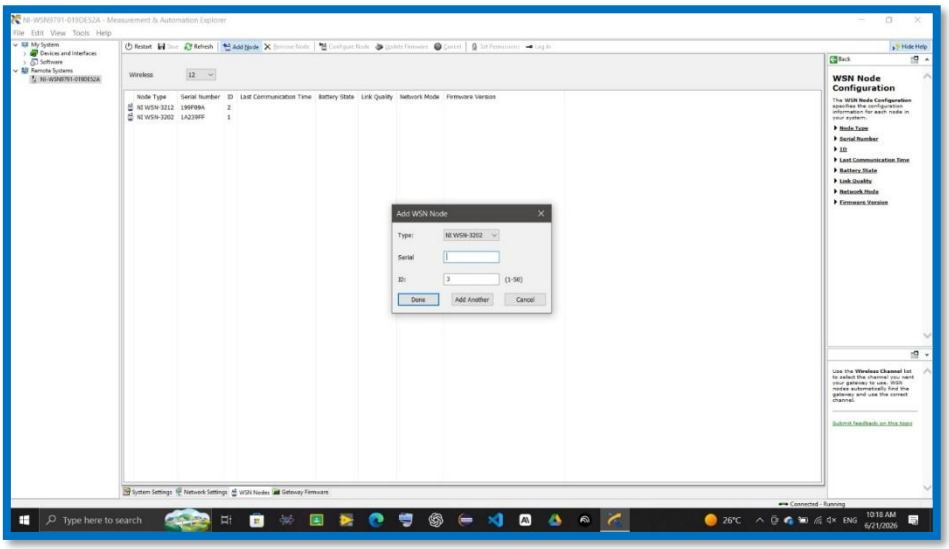


*Figure 3.34: **WSN Node Configuration in NI MAX.***

4.15.3 LabVIEW Integration

After completing the gateway configuration, the NI WSN-9791 can be integrated into a LabVIEW project for real-time data acquisition.

The integration procedure consists of the following steps:

1. Launch LabVIEW and create a new project.
2. Add the NI WSN-9791 as an Existing Target or Device.
3. Expand the gateway to access the available wireless sensor nodes.
4. Select the required analog input channels (AI0–AI3).
5. Create I/O Variables for the selected channels.
6. Connect the variables to Numeric Indicators or Waveform Charts on the Front Panel.
7. Place the acquisition code inside a While Loop to enable continuous monitoring.
8. Optionally, save the acquired measurements using LabVIEW File I/O functions such as CSV or LVM files.

3.16 Chapter Summary

This chapter presented the LabVIEW development environment and explained the fundamental concepts required for building virtual instrumentation applications. It covered front panels, block diagrams, debugging techniques, data structures, graphs and charts, file I/O operations, and the integration of the NI WSN-9791 wireless sensor gateway with LabVIEW for monitoring and data acquisition applications.

CHAPTER

4

Project Problem Formulation

Environmental Monitoring | Wireless Sensor Network | Real-time Data Acquisition | Intelligent Decision Making | Cloud Storage & Synchronization | Early Fire Detection | Rapid Alerts & Response

4.1 Introduction

Forest fires represent one of the most destructive environmental disasters worldwide. They cause severe ecological damage, loss of biodiversity, economic losses, and threats to human life. The rapid spread of fire in forest environments often makes early detection difficult using conventional monitoring methods such as manual patrols, watchtowers, and satellite observation systems. These traditional approaches may suffer from limited coverage, delayed response, and high operational costs.

Recent advances in Wireless Sensor Network (WSN) technology have enabled the development of intelligent monitoring systems capable of continuously observing environmental conditions in real time. By deploying multiple sensor nodes across a protected area, critical environmental parameters can be measured, transmitted wirelessly, analyzed automatically, and used to generate early warnings before a fire develops into a large-scale disaster.

The objective of this project is to design and implement a Wireless Sensor Network-Based Forest Fire Detection and Environmental Monitoring System capable of monitoring environmental conditions and assessing fire risk levels in real time. The proposed system integrates National Instruments Wireless Sensor Network devices, environmental sensors, LabVIEW software, Python-based decision-making algorithms, Firebase cloud services, and automated notification mechanisms to create a complete early warning platform.

The system continuously acquires measurements from multiple environmental sensors including soil moisture, carbon monoxide concentration, carbon dioxide concentration, and temperature sensors. These measurements are processed using LabVIEW and transmitted to a Python-based classification engine that evaluates environmental conditions using a multi-level danger assessment algorithm. Depending on the severity of the detected conditions, the system can generate alerts through cloud services, e-mail notifications, and automated phone calls.

Unlike conventional fire detection systems that react only after visible smoke or flames appear, the proposed system focuses on identifying early warning indicators associated with fire development. These indicators include elevated temperature levels, increased concentrations of combustion gases, and critically low soil moisture conditions. By combining multiple environmental parameters simultaneously, the system improves detection reliability while reducing the probability of false alarms.

The remainder of this chapter presents the problem formulation, system requirements, proposed architecture, sensor selection criteria, fire detection methodology, danger classification strategy, and the overall design approach adopted for the implementation of the proposed forest fire detection system.

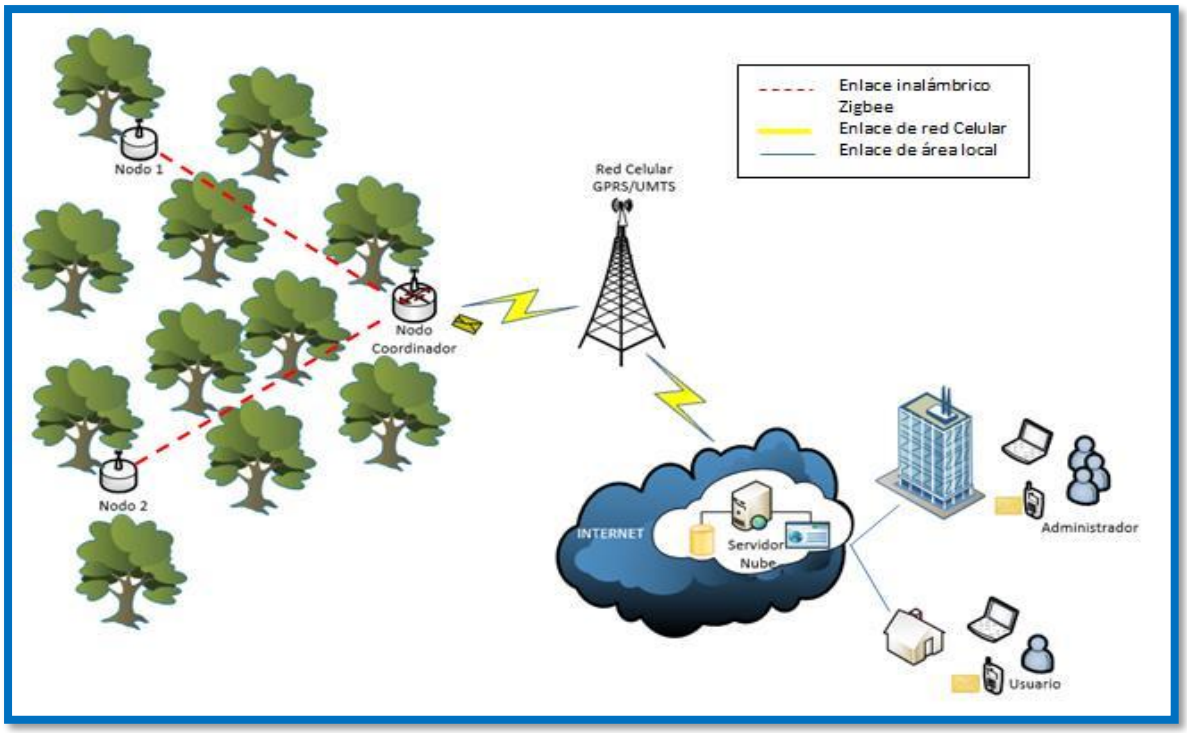


Figure 4.1: ***General architecture of a Wireless Sensor Network (WSN)-based forest fire monitoring system.***

4.2 Forest Fire Problem Analysis

Forest fires represent a significant environmental challenge due to their ability to spread rapidly over large geographical areas. Once ignition occurs, factors such as high temperature, dry vegetation, low soil moisture, wind speed, and the accumulation of combustible gases can accelerate fire propagation, making containment increasingly difficult. As a result, delays in detection often lead to extensive environmental damage, destruction of wildlife habitats, economic losses, and threats to nearby communities.

The fire development process typically progresses through several stages. During the initial stage, environmental conditions become favorable for ignition due to elevated temperatures and reduced moisture levels. As combustion begins, gases such as carbon monoxide (CO) and carbon dioxide (CO_2) are released into the surrounding atmosphere. During the growth stage, temperature rises significantly, gas concentrations increase, and the fire expands to nearby vegetation. If the fire remains undetected, it eventually reaches a fully developed stage characterized by intense heat generation, dense smoke production, and widespread environmental destruction.

One of the major challenges associated with forest fire management is the limitation of conventional monitoring techniques. Traditional approaches such as manual inspection, watchtowers, and satellite imaging may suffer from delayed response times and insufficient monitoring coverage, particularly in remote forest regions. Consequently, there is a need for automated systems capable of continuously monitoring environmental conditions and identifying fire indicators before visible flames appear.

To address this challenge, the proposed system focuses on monitoring four critical environmental parameters that are strongly associated with fire development: temperature, soil moisture, carbon monoxide concentration, and carbon dioxide concentration. By analyzing these parameters simultaneously, the system can identify abnormal environmental conditions at an early stage and provide timely alerts before a fire evolves into a large-scale disaster.

California wildfires have demonstrated the devastating consequences of delayed fire detection, resulting in extensive environmental damage, destruction of infrastructure, and significant economic losses. These incidents highlight the need for intelligent real-time monitoring systems capable of identifying fire indicators at an early stage.

***Figure 4.2:** Aerial view of a California wildfire illustrating the rapid spread and large-scale environmental impact of forest fires.*

4.3 Existing Fire Detection Methods

Several techniques have been used for forest fire monitoring and detection over the years. These methods aim to identify fire incidents and provide alerts to emergency response teams. However, many traditional approaches suffer from limitations related to coverage, response time, operational cost, and detection accuracy.

One of the oldest monitoring methods is the use of observation towers, where personnel continuously monitor forest areas and report the presence of smoke or fire. Although this approach is relatively simple, its effectiveness depends heavily on human observation and environmental visibility conditions. Factors such as fog, darkness, and large monitoring areas can significantly reduce detection efficiency.

Satellite-based monitoring systems are also widely used for wildfire detection. These systems can cover large geographical regions and provide valuable information about active fires. However, satellite imagery is often limited by revisit time, cloud coverage, image resolution, and processing delays, which may prevent the early detection of small fire events.

Aerial monitoring using aircraft and drones provides higher observation accuracy and improved mobility. Nevertheless, these systems require significant operational resources, trained personnel, and continuous maintenance, resulting in relatively high operational costs.

To overcome these limitations, Wireless Sensor Networks (WSNs) have emerged as an effective solution for environmental monitoring and early fire detection. WSN-based systems continuously monitor environmental parameters directly inside the protected area and can identify abnormal conditions before visible flames appear. This capability enables earlier intervention, faster response times, and improved reliability compared to conventional monitoring techniques.

***Table 4.1:* Comparison of Forest Fire Detection Methods**

Method	***Advantages***	***Limitations***
Observation Towers	*Simple and low installation cost*	*Human-dependent and limited coverage*
Satellite Monitoring	*Large geographical coverage*	*Delayed detection and weather dependency*
Aerial Monitoring	*High observation accuracy*	*High operational cost*
Wireless Sensor Networks	*Real-time monitoring and early detection*	*Requires sensor deployment and network infrastructure*

The limitations of conventional fire monitoring techniques motivate the development of the proposed Wireless Sensor Network-based forest fire detection system presented in this project.

Figure 4.3: Traditional forest fire monitoring using observation towers.

4.4 Problem Statement

Forest fires often remain undetected during their initial stages, allowing combustion to spread rapidly before appropriate countermeasures can be implemented. The delay between fire ignition and detection significantly increases the risk of environmental destruction, loss of natural resources, economic damage, and threats to human safety. In many forest environments, traditional monitoring approaches are unable to provide continuous real-time observation, making early intervention difficult.

Existing fire detection systems typically rely on the observation of visible flames, smoke plumes, or thermal anomalies. However, these indicators often appear after the fire has already developed beyond its initial stage. Consequently, response teams may receive warnings too late to effectively prevent fire propagation.

Furthermore, environmental factors associated with fire development, such as elevated temperature, increased concentrations of combustion gases, and reduced soil moisture levels, are frequently ignored by conventional monitoring systems. These parameters can provide valuable early indicators of potential fire occurrence before visible signs become apparent.

Therefore, there is a need for an intelligent monitoring system capable of continuously acquiring environmental measurements, analyzing multiple fire-related indicators simultaneously, and generating real-time alerts whenever hazardous conditions are detected. Such a system should operate autonomously, provide reliable communication, minimize false alarms, and support remote monitoring capabilities.

To address these challenges, this project proposes a Wireless Sensor Network-based forest fire detection system that integrates environmental sensors, National Instruments wireless acquisition devices, LabVIEW processing, Python-based decision-making algorithms, Firebase cloud services, and automated notification mechanisms. The system continuously monitors temperature, soil moisture, carbon monoxide (CO), and carbon dioxide (CO_2) levels to assess environmental conditions and determine fire risk levels in real time.

The proposed solution aims to provide early fire detection, improve emergency response time, reduce potential environmental losses, and enhance the overall reliability of forest monitoring operations.

Figure 4.4: Concept of early forest fire detection using environmental monitoring systems.

4.5 System Requirements

The successful implementation of the proposed forest fire detection system requires a set of functional and non-functional requirements to ensure reliable operation, accurate fire detection, real-time monitoring, and efficient communication between all system components. These requirements define the expected behavior and performance characteristics of the proposed system.

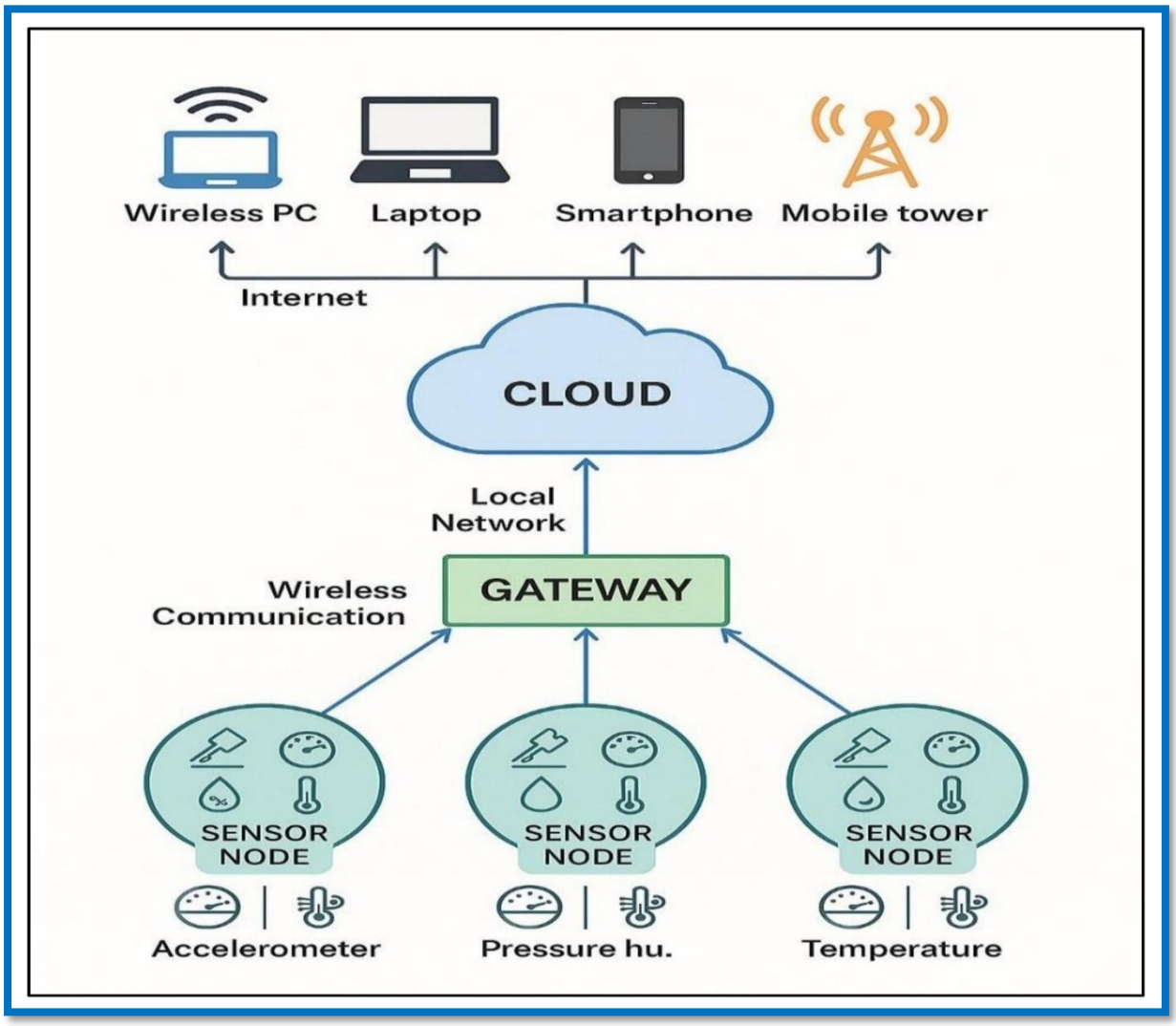


Figure 4.5: ***General requirements of a real-time environmental monitoring system including sensing, wireless communication, data processing, cloud connectivity, and alert generation.***

4.5.1 Functional Requirements

Functional requirements describe the essential operations that must be performed by the system to achieve its intended objectives.

Continuous Environmental Monitoring

The system shall continuously monitor environmental conditions by acquiring measurements from multiple sensors deployed within the monitored area. The selected environmental parameters include:

- Soil Moisture
- Carbon Monoxide (CO)
- Carbon Dioxide (CO_2)
- Temperature

These parameters were selected because they represent critical indicators associated with forest fire development and environmental dryness.

Wireless Data Acquisition

The system shall utilize National Instruments Wireless Sensor Network devices to acquire sensor measurements and transmit them wirelessly to the central gateway. The wireless architecture eliminates extensive cabling requirements and supports remote deployment in forest environments.

Real-Time Data Processing

The acquired measurements shall be processed in real time using LabVIEW software. Sensor values must be converted into engineering units, displayed through the monitoring interface, and prepared for further analysis by the decision-making algorithm.

Fire Risk Evaluation

The system shall evaluate environmental conditions using a multi-parameter classification algorithm developed in Python. The algorithm analyzes all sensor measurements simultaneously and assigns an appropriate danger level based on predefined thresholds and decision rules.

Cloud-Based Monitoring

The system shall upload real-time measurements and classification results to Firebase Realtime Database to enable remote monitoring and data accessibility from external devices.

Alert Generation

When abnormal environmental conditions are detected, the system shall automatically generate warning notifications. Alert mechanisms include:

- Email Notifications
- Automated Voice Calls
- Dashboard Alarm Indicators

These notifications ensure rapid response to potential fire incidents.

4.5.2 Non-Functional Requirements

Non-functional requirements define the quality attributes and performance expectations of the system.

Reliability

The system must provide reliable operation under continuous monitoring conditions and maintain stable communication between sensor nodes, gateway devices, and processing units.

Accuracy

Sensor measurements and classification results must accurately represent actual environmental conditions to minimize false alarms and missed detections.

Real-Time Performance

The system should process incoming sensor data with minimal delay to ensure timely detection of hazardous environmental conditions.

Scalability

The proposed architecture should support the addition of future sensor nodes and environmental parameters without requiring major modifications to the system structure.

Maintainability

The software architecture should remain modular and easy to maintain, allowing future improvements and updates to be implemented efficiently.

Availability

The monitoring platform should remain continuously operational and capable of providing uninterrupted environmental monitoring and alert generation services.

4.6 Proposed Solution

To address the limitations of conventional forest fire monitoring techniques, a Wireless Sensor Network (WSN)-based environmental monitoring and fire detection system is proposed. The system is designed to continuously acquire environmental measurements, analyze fire-related indicators, classify risk levels, and generate real-time notifications whenever hazardous conditions are detected.

The proposed system consists of four major layers: the sensing layer, data acquisition layer, decision-making layer, and notification layer. These layers operate together to provide continuous environmental monitoring and early fire detection capabilities.

Sensing Layer

The sensing layer is responsible for collecting environmental measurements from the monitored area. Four sensors are utilized in the proposed system:

- Soil Moisture Sensor
- MQ-135 Carbon Dioxide Sensor
- MQ-7 Carbon Monoxide Sensor
- K-Type Thermocouple Temperature Sensor

These sensors continuously monitor environmental conditions associated with forest fire development and provide real-time measurements to the wireless acquisition nodes.

Data Acquisition Layer

Sensor measurements are acquired using National Instruments wireless sensor network devices. The NI WSN-3202 and NI WSN-3212 nodes receive analog sensor signals and transmit the collected data wirelessly to the NI WSN-9791 Gateway. The gateway serves as the central communication unit of the network and transfers the acquired measurements to the host computer through an Ethernet connection.

Decision-Making Layer

After data acquisition, LabVIEW performs signal conditioning, calibration, scaling, and visualization of sensor measurements. The processed data is then transmitted to a Python-based classification algorithm through a TCP communication channel. The Python module evaluates the environmental conditions using predefined thresholds and multi-parameter decision logic to determine the current fire risk level.

The classification algorithm categorizes environmental conditions into five levels:

- Normal
- Low Warning
- Medium Danger
- High Danger
- Critical

This multi-level approach improves decision accuracy and reduces false alarms compared to single-sensor detection systems.

Notification Layer

When abnormal environmental conditions are detected, the system automatically generates warning notifications. Sensor measurements and classification results are uploaded to Firebase Realtime Database for remote monitoring. In addition, warning e-mails and automated voice calls are generated whenever dangerous environmental conditions are detected.

The integration of wireless sensing, real-time processing, cloud connectivity, and automated alert generation enables the proposed system to provide an effective and reliable solution for early forest fire detection.

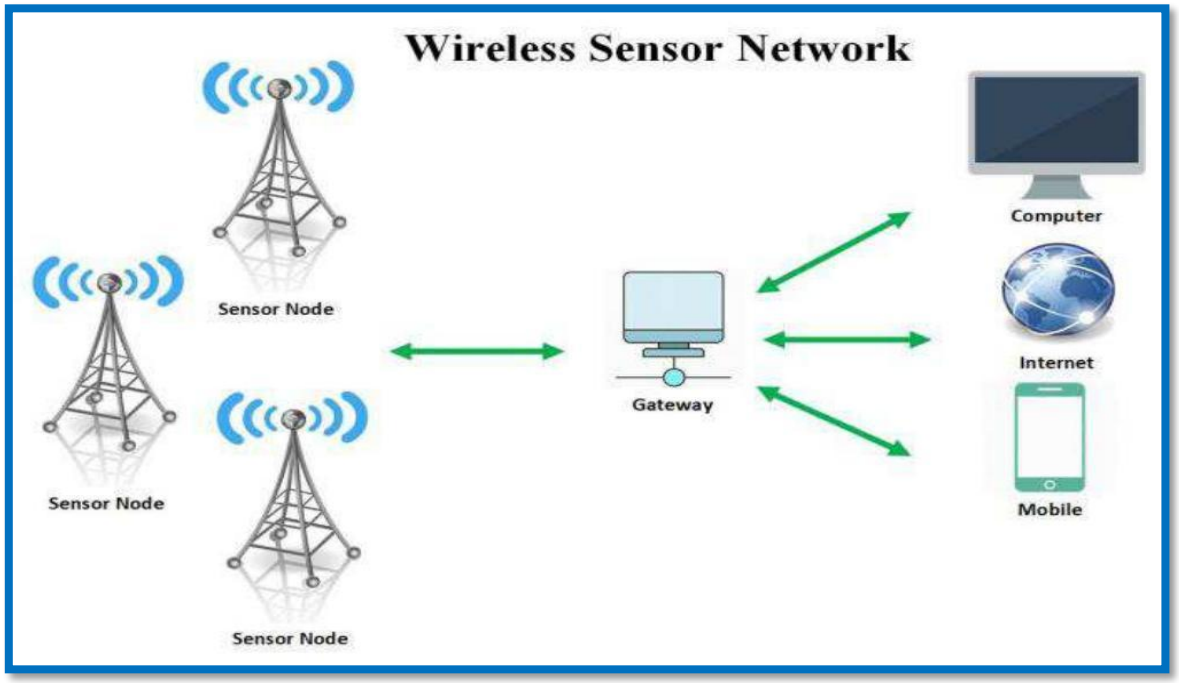


Figure 4.6: General architecture of the proposed Wireless Sensor Network-based forest fire detection system.

4.7 System Architecture

The proposed forest fire detection system follows a layered architecture that integrates environmental sensing, wireless communication, real-time data processing, cloud connectivity, and automated notification services. The architecture was designed to provide continuous monitoring of forest conditions while ensuring reliable data transmission and rapid response to potential fire incidents.

The system begins with the sensing layer, where environmental sensors continuously measure critical parameters associated with forest fire development. These parameters include soil moisture, carbon monoxide concentration, carbon dioxide concentration, and temperature. Each sensor produces an analog signal proportional to the measured environmental condition.

The acquired signals are connected to National Instruments Wireless Sensor Network nodes. The NI WSN-3202 node is responsible for acquiring measurements from the Soil Moisture Sensor and MQ-135 sensor, while the NI WSN-3212 node acquires temperature measurements from the K-Type Thermocouple and gas concentration data from the MQ-7 sensor. The wireless nodes perform local data acquisition and transmit the measurements wirelessly using the IEEE 802.15.4 communication protocol.

All transmitted measurements are received by the NI WSN-9791 Gateway, which acts as the central communication unit of the network. The gateway collects data from all wireless nodes and forwards the information to a host computer through an Ethernet connection. This architecture eliminates the need for extensive wiring and allows flexible deployment in large forest environments.

At the monitoring station, LabVIEW is used to perform data acquisition, calibration, signal conditioning, visualization, and data logging. Sensor readings are displayed through a graphical user interface that enables real-time observation of environmental conditions. The processed measurements are then transmitted to a Python-based decision-making module using a TCP communication channel.

The Python module serves as the intelligent decision-making layer of the system. It evaluates sensor measurements using a multi-parameter classification algorithm and determines the current fire danger level. The algorithm classifies environmental conditions into five levels: Normal, Low Warning, Medium Danger, High Danger, and Critical. The resulting classification is transmitted back to LabVIEW and simultaneously uploaded to Firebase Realtime Database for remote monitoring.

Finally, the notification layer generates automatic alerts whenever dangerous environmental conditions are detected. Firebase provides cloud-based monitoring, while e-mail notifications and automated voice calls ensure that responsible personnel receive timely warnings and can take appropriate action before a fire escalates.

The adopted architecture provides a complete end-to-end solution for environmental monitoring, fire risk assessment, cloud integration, and emergency notification, making it suitable for early forest fire detection applications.

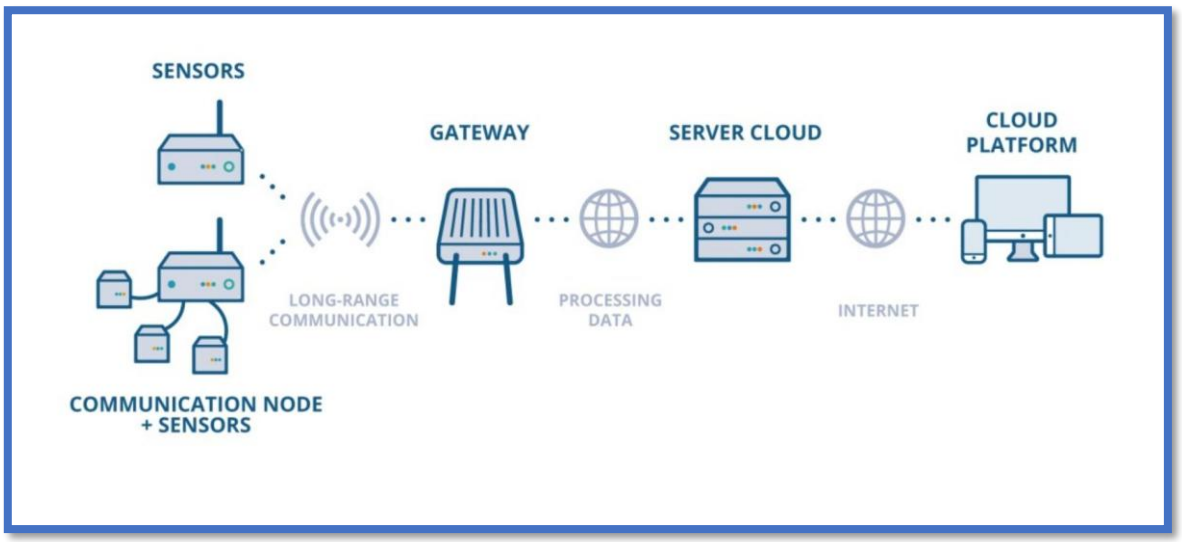


***Figure 4.7:* Architecture of a Wireless Sensor Network-based environmental monitoring system.**

4.8 Soil Moisture Sensor

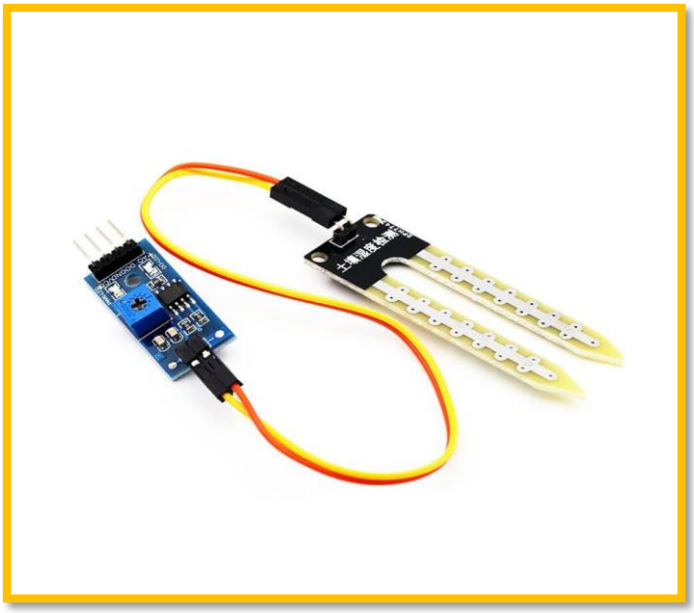


Figure 4.8: Soil Moisture Sensor used for measuring the water content of soil.

4.8.1 Introduction

Soil moisture is one of the most important environmental parameters affecting forest fire occurrence. The amount of water stored within the soil directly influences the moisture content of vegetation, leaf litter, and organic materials present on the forest floor. When soil moisture decreases significantly, surrounding vegetation becomes dry and highly susceptible to ignition. Therefore, monitoring soil moisture provides an effective method for assessing fire risk before the appearance of flames or smoke.

In the proposed system, the Soil Moisture Sensor is utilized as an early fire risk indicator. Rather than detecting an active fire, the sensor evaluates environmental dryness conditions that may increase the probability of fire occurrence. This enables the system to provide preventive warnings before hazardous conditions develop into an actual fire event.

4.8.2 Operating Principle

The Soil Moisture Sensor operates based on the variation of soil electrical conductivity with moisture content. The sensing probe consists of two conductive electrodes inserted into the soil. Since water improves electrical conductivity, wet soil exhibits lower electrical resistance compared to dry soil.

When the soil contains a high amount of water, electrical conductivity increases and the sensor output voltage decreases. Conversely, when the soil becomes dry, electrical resistance increases, resulting in a higher output voltage. The sensor converts this variation into an analog voltage signal that can be measured and processed by the data acquisition system.

4.8.3 Technical Specifications

Table 4.2 summarizes the main technical specifications of the Soil Moisture Sensor used in this project.

Parameter	Specification
Operating Voltage	3.3 – 5.5 V DC
Output Type	Analog Voltage
Output Range	1.17 – 4.96 V
Current Consumption	Approximately 35 mA
Acquisition Node	NI WSN-3202
Input Channel	AI0

4.8.4 Hardware Integration

The Soil Moisture Sensor is connected directly to the NI WSN-3202 wireless acquisition node. The sensor output terminal (A0) is connected to analog input channel AI0, while the VCC and GND terminals are connected to the node power supply and common ground respectively.

The NI WSN-3202 digitizes the analog voltage signal using its internal Analog-to-Digital Converter (ADC) and transmits the acquired measurements wirelessly to the NI WSN-9791 Gateway through the IEEE 802.15.4 wireless communication protocol.

4.8.5 Calibration and Data Conversion

The sensor produces an analog voltage that must be converted into a meaningful moisture percentage. Based on experimental calibration, the following reference points were established:

- Dry Soil Voltage = 5.1 V
- Fully Saturated Soil Voltage = 1.15 V

The moisture percentage is calculated using the following calibration equation:

$$Moisture(\%) = \frac{(5.1 - V_{in})}{(5.1 - 1.15)} \times 100$$

where:

- V_{in}= Sensor output voltage (V)
- Moisture (%) = Calculated soil moisture percentage

This equation was implemented in LabVIEW to convert raw sensor voltages into real-time moisture measurements.

4.8.6 Moisture-Based Risk Classification

To support fire risk assessment, soil moisture measurements are classified into four operating zones.

*Table 4.3: **Moisture-Based Risk status.***

Moisture Range (%)	**Status**
> 60%	Safe
40% – 60%	Low Warning
20% – 40%	Warning
< 20%	High Warning

Low moisture levels indicate dry environmental conditions and increase the probability of vegetation ignition. Therefore, soil moisture serves as a primary indicator in the fire risk assessment process.

4.8.7 Role in Fire Detection System

Within the proposed fire detection algorithm, soil moisture measurements contribute to the overall danger assessment process. Extremely low moisture values increase the fire danger level, particularly when combined with elevated temperature or abnormal gas concentrations. By monitoring environmental dryness continuously, the Soil Moisture Sensor provides an important early-warning mechanism that helps identify hazardous conditions before the occurrence of an actual fire.

4.9 MQ-135 Air Quality and Carbon Dioxide Sensor

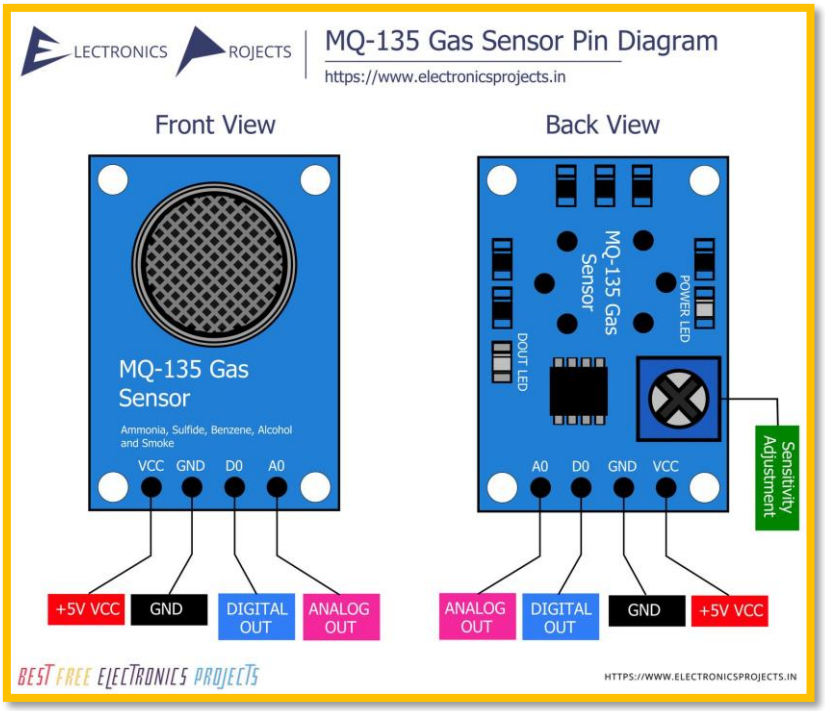


*Figure 4.9: **MQ-135 gas sensor used for air quality monitoring and carbon dioxide detection.***

4.9.1 Introduction

The MQ-135 gas sensor is employed in the proposed forest fire detection system to monitor air quality degradation and detect abnormal increases in carbon dioxide (CO_2) concentration. During the early stages of combustion, particularly during the smoldering phase, vegetation releases significant amounts of carbon dioxide and other gaseous by-products before visible flames appear. Therefore, monitoring CO_2 concentration provides an effective method for identifying potential fire conditions at an early stage.

In addition to carbon dioxide detection, the MQ-135 sensor is capable of detecting smoke, ammonia, benzene, alcohol vapors, and several volatile organic compounds (VOCs). This multi-gas sensitivity makes the sensor suitable for environmental monitoring applications where air quality assessment is required

4.9.2 Operating Principle

The MQ-135 operates using a tin dioxide (SnO_2) semiconductor sensing element. Under normal air conditions, oxygen molecules are adsorbed onto the sensor surface, increasing its internal resistance. When reducing gases such as smoke or combustion gases are present,

chemical reactions occur at the sensor surface, releasing trapped electrons and decreasing the sensor resistance.

As the concentration of target gases increases, the internal resistance decreases and the output voltage rises. This voltage variation is then converted into a gas concentration value through a calibration process implemented in LabVIEW.

4.9.3 Technical Specifications

Table 4.4 presents the main technical specifications of the MQ-135 sensor.

Parameter	**Specification**
Sensor Model	MQ-135
Operating Voltage	5 V DC
Heating Voltage	5 V DC
Detectable Gases	CO_2, Smoke, NH_3, Benzene, Alcohol
Detection Range	10 – 10000 ppm
Acquisition Node	NI WSN-3202
Input Channel	AI3

4.9.4 Hardware Integration

The MQ-135 sensor is interfaced with the NI WSN-3202 wireless acquisition node. The analog output pin (AOUT) of the sensor is connected to analog input channel AI3 of the NI WSN-3202 node. The sensor is powered using a regulated 5 V supply to ensure stable operation of the sensing and heating elements.

After digitization by the NI WSN-3202 node, the acquired measurements are transmitted wirelessly to the NI WSN-9791 Gateway using the IEEE 802.15.4 wireless communication protocol. The gateway then forwards the measurements to the LabVIEW monitoring station for processing and visualization.

4.9.5 Calibration and CO_2 Concentration Calculation

The analog voltage generated by the MQ-135 sensor is converted into carbon dioxide concentration (ppm) using a two-point linear calibration model implemented in LabVIEW.

The calibration reference points are:

- 400 ppm → 0.12 V
- 10000 ppm → 3.0 V

The conversion equation is:

$$CO_2(ppm) = \left(\frac{10000 - 400}{3.0 - 0.12}\right)(V_{in} - 0.12) + 400$$

where:

- V_{in}= Sensor output voltage (V)
- CO_2= Carbon dioxide concentration (ppm)

This calibration allows the system to convert raw voltage readings into meaningful environmental measurements.

4.9.6 Fire Detection Threshold

Under normal forest conditions, ambient carbon dioxide concentrations typically remain below the critical danger threshold. Experimental analysis showed that CO_2 concentrations exceeding 2000 ppm indicate abnormal environmental conditions associated with combustion activities.

Therefore, the proposed system defines:

$$CO_2 > 2000\ ppm$$

as a critical fire indicator.

When this threshold is exceeded, the Python classification algorithm immediately elevates the system danger level to **CRITICAL (Level 4)** and initiates emergency notifications.

4.9.7 Role in Fire Detection System

The MQ-135 sensor serves as one of the primary fire detection sensors within the proposed system. Its ability to detect elevated carbon dioxide levels and smoke-related gases enables the identification of combustion processes during their early stages. Unlike temperature-based detection alone, gas concentration monitoring provides additional evidence of combustion activity and improves overall system reliability.

The sensor output is continuously evaluated by the Python-based decision-making algorithm and combined with temperature, carbon monoxide, and soil moisture measurements to determine the overall fire danger level. This multi-parameter approach significantly reduces false alarms and enhances detection accuracy.

4.10 MQ-7 Carbon Monoxide Sensor

***Figure 4.10*: MQ-7 carbon monoxide gas sensor used for combustion gas detection.**

4.10.1 Introduction

Carbon monoxide (CO) is one of the primary gases produced during incomplete combustion processes. During the early stages of forest fires, particularly during smoldering combustion, significant amounts of carbon monoxide are released before the appearance of visible flames. Consequently, monitoring carbon monoxide concentration provides an effective approach for detecting fire-related activities at an early stage.

In the proposed system, the MQ-7 sensor is utilized to continuously monitor carbon monoxide concentration and identify combustion-related environmental changes. The sensor serves as one of the primary indicators used by the fire detection algorithm to classify danger levels and determine the likelihood of fire occurrence.

4.10.2 Operating Principle

The MQ-7 sensor is based on a tin dioxide (SnO_2) semiconductor sensing element. The electrical resistance of the sensing material changes according to the concentration of carbon monoxide present in the surrounding environment.

When carbon monoxide concentration increases, chemical reactions occur on the sensor surface, resulting in a decrease in the internal resistance of the sensing layer. This resistance variation produces a corresponding change in the analog output voltage, which is subsequently processed and converted into carbon monoxide concentration values.

The MQ-7 sensor employs a cyclic heating mechanism consisting of alternating high-temperature and low-temperature operating periods to improve selectivity toward carbon monoxide and reduce interference from other gases.

4.10.3 Technical Specifications

***Table 4.5:* Summarizes the technical specifications of the MQ-7 sensor.**

Parameter	***Specification***
Sensor Type	*Semiconductor Gas Sensor*
Target Gas	*Carbon Monoxide (CO)*
Sensing Material	*Tin Dioxide (SnO_2)*
Detection Range	*20 – 2000 ppm*
Operating Voltage	*5 V DC*
Heater Voltage	*5 V / 1.4 V Cyclic Operation*
Heater Resistance	*33 Ω ± 5%*
Power Consumption	*< 350 mW*
Acquisition Channel	*AI1*

4.10.4 Hardware Integration

The analog output of the MQ-7 sensor is connected to analog input channel AI1 of the National Instruments wireless acquisition system. The NI WSN node samples the analog voltage generated by the sensor and converts it into digital data for wireless transmission.

The acquired measurements are transmitted to the NI WSN-9791 Gateway and subsequently forwarded to the LabVIEW monitoring station, where signal scaling, visualization, and fire detection processing are performed.

4.10.5 Calibration and CO Concentration Calculation

To convert the raw analog voltage into carbon monoxide concentration, a two-point linear calibration model was implemented within LabVIEW.

The calibration reference points are:

- 0.5 V → 0 ppm CO
- 4.5 V → 500 ppm CO

The concentration is calculated using the following equation:

$$CO(ppm) = \frac{(V_{in} - 0.5)}{(4.5 - 0.5)} \times 500$$

where:

- V_{in}= Sensor output voltage (V)
- CO= Carbon monoxide concentration (ppm)

This scaling equation enables continuous real-time monitoring of carbon monoxide levels within the protected environment.

4.10.6 Smoldering Combustion Classification

Carbon monoxide measurements are classified into multiple combustion stages to support fire detection.

***Table 4.6:** presents the classification criteria implemented in the proposed system.*

CO Concentration (ppm)	***Classification***
≤ 25	*Normal*
25 – 100	*Light Smoldering Combustion*
> 100	*Heavy Smoldering Combustion*

The Light Smoldering stage indicates the possible initiation of combustion activities, whereas the Heavy Smoldering stage represents a significantly higher probability of active fire development.

4.10.7 Role in Fire Detection Algorithm

The MQ-7 sensor plays a critical role in the Python-based decision-making algorithm. Carbon monoxide concentration is one of the primary parameters used for danger level classification.

According to the implemented algorithm:

- CO > 25 ppm activates the Light Smoldering indicator.
- CO > 100 ppm activates the Heavy Smoldering indicator.
- Heavy smoldering conditions combined with elevated temperature levels may directly trigger CRITICAL fire status.
- Heavy smoldering conditions combined with dry soil may generate HIGH DANGER classification.

These decision rules allow the system to identify combustion processes before the appearance of visible flames, thereby improving early fire detection performance.

4.10.8 Advantages of Using MQ-7 in Forest Fire Detection

The MQ-7 sensor offers several advantages in forest fire monitoring applications:

- Early detection of combustion activities.
- High sensitivity to carbon monoxide.
- Low implementation cost.
- Simple hardware integration.
- Improved detection reliability when combined with temperature and CO_2 measurements.

For these reasons, the MQ-7 sensor was selected as a core component of the proposed fire detection system.

4.11 K-Type Thermocouple Temperature Sensor

Figure 4.11: ***K-Type thermocouple temperature sensor used for environmental temperature monitoring.***

4.11.1 Introduction

Temperature is one of the most significant indicators associated with forest fire development. During the ignition and smoldering stages, the surrounding temperature begins to increase due to heat released from combustion processes. Therefore, continuous temperature monitoring is essential for identifying abnormal environmental conditions and assessing fire risk levels.

In the proposed system, a K-Type Thermocouple is employed to provide accurate temperature measurements over a wide operating range. Compared to conventional temperature sensors, thermocouples offer higher temperature tolerance, improved durability, and suitability for harsh outdoor environments, making them ideal for forest fire monitoring applications

4.11.2 Operating Principle

The K-Type Thermocouple operates based on the **Seebeck Effect**, where a small electrical voltage is generated when two dissimilar metallic conductors are joined together and exposed to a temperature difference.

The thermocouple consists of:

- Chromel (Nickel-Chromium Alloy)
- Alumel (Nickel-Aluminum Alloy)

When the measuring junction experiences a temperature change, a thermoelectric voltage is generated. This voltage is proportional to the temperature difference between the sensing junction and the reference junction. The generated voltage is then measured and converted into temperature values by the acquisition system

4.11.3 Technical Specifications

*Table 4.7: **Summarizes the specifications of the K-Type Thermocouple used in the proposed system**.*

Parameter	**Specification**
Thermocouple Type	Type K
Conductors	Chromel / Alumel
Part Number	745690-K002
Temperature Range	-200°C to 482°C
Cable Length	2 m
Insulation Material	Fiberglass
Certification	CE, RoHS
Acquisition Node	NI WSN-3212

4.11.4 Hardware Integration

The thermocouple is directly connected to the NI WSN-3212 Wireless Thermocouple Node. Unlike conventional analog sensors, thermocouples generate very small voltage signals in the microvolt range, requiring high-resolution signal acquisition hardware.

The NI WSN-3212 provides:

- High-resolution analog acquisition.
- Cold Junction Compensation (CJC).
- Temperature conversion.
- Wireless transmission to the gateway.

The measured temperature values are transmitted wirelessly to the NI WSN-9791 Gateway and then forwarded to LabVIEW for visualization and processing.

4.11.5 Environmental Considerations

Several environmental factors can influence thermocouple performance:

High Temperature Exposure

Although the sensing junction can withstand very high temperatures, prolonged exposure beyond the insulation limit may damage the fiberglass jacket and reduce measurement accuracy.

Humidity and Moisture

Moisture absorption within the insulation material may introduce measurement errors and signal drift. Therefore, proper environmental protection should be considered for long-term outdoor deployment.

Mechanical Stress

Excessive bending or vibration may damage the sensing wires and affect measurement reliability. Appropriate cable support and strain-relief mechanisms are recommended.

4.11.6 Temperature Threshold Classification

The Python fire detection algorithm classifies temperature measurements into several operating zones.

Table 4.8: ***presents the implemented temperature classification criteria.***

Temperature Range	***Classification***
≤ 20°C	*Normal*
20°C – 40°C	*Temperature Warning*
40°C – 60°C	*High Temperature Warning*
> 60°C	*Critical Temperature*

The critical threshold of 60°C is treated as a strong fire indicator because such temperatures are rarely observed under normal environmental conditions.

4.11.7 Role in Fire Detection Algorithm

Temperature measurements are continuously evaluated by the Python-based classification algorithm.

According to the implemented logic:

- Temperatures above 20°C contribute to warning conditions.
- Temperatures above 40°C increase the danger level.
- Temperatures exceeding 60°C immediately trigger **CRITICAL** status.
- Elevated temperatures combined with abnormal gas concentrations significantly increase the probability of active combustion.

The temperature sensor therefore acts as one of the primary parameters used to determine the overall fire danger level and improve detection reliability.

4.11.8 Advantages of Using K-Type Thermocouple

The K-Type Thermocouple offers several advantages for forest fire monitoring applications:

- Wide temperature measurement range.
- High thermal stability.
- Fast response time.

- Suitability for harsh environmental conditions.
- High reliability when integrated with NI WSN acquisition devices.

These characteristics make the K-Type Thermocouple an effective temperature sensing solution within the proposed forest fire detection system.

4.12 Fire Detection Algorithm

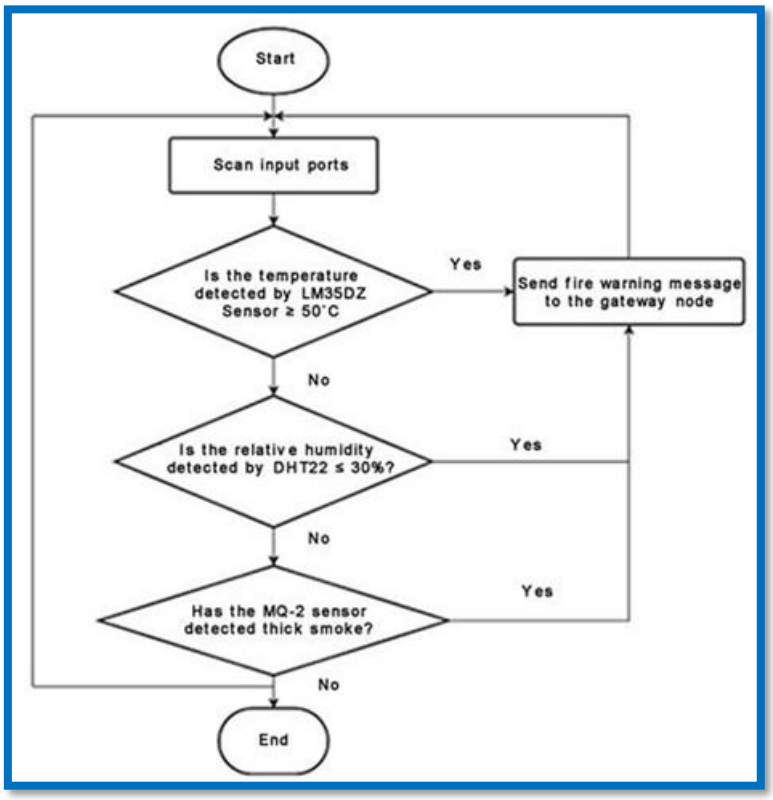


Figure 4.12: General flowchart of the fire detection and danger classification algorithm.

4.12.1 Introduction

The core intelligence of the proposed forest fire monitoring system is implemented through a Python-based fire detection algorithm. Unlike conventional systems that rely on a single sensor parameter, the proposed algorithm performs multi-parameter analysis by simultaneously evaluating soil moisture, carbon monoxide concentration, carbon dioxide concentration, and temperature measurements. This approach improves detection reliability and reduces false alarms by considering the interaction between multiple environmental indicators rather than relying on a single measurement.

The algorithm receives sensor data from LabVIEW through a TCP communication channel. After receiving the measurements, the data is validated, analyzed, classified, and assigned a danger level ranging from 0 to 4. The resulting classification is then returned to LabVIEW, uploaded to Firebase Realtime Database, and used to trigger notification mechanisms when necessary.

4.12.2 Data Processing Sequence

The fire detection process consists of several consecutive stages:

1. Sensor data acquisition from NI WSN devices.
2. Signal conditioning and scaling in LabVIEW.
3. TCP transmission from LabVIEW to Python.
4. Sensor value validation.
5. Fire risk analysis and classification.
6. Danger level generation.
7. Firebase database update.
8. Notification dispatch if required.

This sequence ensures that all sensor readings are processed consistently before any decision is made.

4.12.3 Sensor Validation

Before executing the classification process, all incoming sensor values are validated to ensure data integrity. The algorithm checks for:

- Missing values.
- NaN (Not a Number) values.
- Infinite values.
- Communication faults.

Invalid readings are automatically replaced with safe fallback values to prevent false alarms and system crashes. Additionally, sensor fault notifications can be generated if connectivity problems are detected.

This validation stage increases system reliability and ensures continuous operation even when individual sensors experience temporary failures.

4.12.4 Multi-Parameter Decision Logic

The proposed algorithm evaluates four primary environmental indicators:

Temperature Analysis

Temperature measurements are analyzed to identify abnormal heat conditions associated with combustion processes.

Soil Moisture Analysis

Soil moisture readings are used to estimate environmental dryness and vegetation susceptibility to ignition.

Carbon Monoxide Analysis

Carbon monoxide concentration is evaluated to detect smoldering combustion and incomplete burning processes.

Carbon Dioxide Analysis

Carbon dioxide concentration is monitored to identify combustion-related gas emissions and abnormal air quality conditions.

Each parameter generates one or more logical indicators that are subsequently combined within the decision-making process.

4.12.5 Decision Strategy

The classification algorithm follows a priority-based decision structure. Higher danger levels are evaluated before lower danger levels to ensure that the most severe condition is always reported.

The evaluation sequence is:

CRITICAL
↓
HIGH DANGER
↓
MEDIUM DANGER
↓
LOW WARNING
↓
NORMAL

As soon as a condition satisfies a higher-priority rule, the corresponding danger level is assigned and the remaining lower-priority conditions are ignored. This strategy guarantees consistent and deterministic classification results.

4.12.6 Advantages of the Proposed Algorithm

The implemented fire detection algorithm provides several advantages compared with conventional threshold-based approaches:

- Multi-sensor decision making.
- Improved fire detection reliability.
- Reduced false alarm probability.
- Early identification of smoldering conditions.
- Automatic danger level classification.
- Real-time operation.
- Integration with cloud monitoring services.
- Automated emergency notification support.

By combining environmental conditions, gas concentrations, and temperature measurements within a unified decision framework, the algorithm can identify hazardous situations more effectively than systems that depend on a single sensor parameter.

4.12.7 Role within the Proposed System

The fire detection algorithm serves as the central decision-making component of the proposed system. All environmental measurements collected by the wireless sensor network are ultimately processed by this algorithm. The generated danger level determines the visual indicators displayed in LabVIEW, the information stored in Firebase, and the activation of e-mail and voice-call notification services.

Therefore, the effectiveness of the entire forest fire monitoring system depends directly on the accuracy and reliability of this decision-making process.

4.13 Fire Risk Classification

*Figure 4.13: **Multi-level fire risk assessment and danger classification concept.***

4.13.1 Introduction

To improve decision-making accuracy and provide appropriate responses to changing environmental conditions, the proposed system employs a five-level fire risk classification strategy. Instead of using a simple binary decision (Fire / No Fire), the system classifies environmental conditions into multiple danger levels representing the severity of the detected situation.

This approach allows the system to identify gradual changes in environmental conditions and provide early warnings before a critical fire event occurs. Consequently, emergency personnel can take preventive actions during the early stages of fire development rather than reacting only after fire ignition.

4.13.2 Classification Levels

The Python-based decision algorithm classifies environmental conditions into five danger levels ranging from Level 0 (Normal) to Level 4 (Critical). Each level represents a different degree of fire risk and is associated with specific environmental conditions and response actions.

Table 4.9: ***Fire Risk Classification Levels***

Level	***Status***	***Description***
0	*NORMAL*	*All environmental parameters remain within safe operating limits.*
1	*LOW WARNING*	*Minor abnormalities detected; monitoring attention required.*
2	*MEDIUM DANGER*	*Multiple warning indicators suggest increasing fire risk.*
3	*HIGH DANGER*	*Strong evidence of combustion activity or hazardous environmental conditions.*
4	*CRITICAL*	*Active fire conditions detected or highly probable fire occurrence.*

4.13.3 Level 0 – NORMAL

The NORMAL state represents safe environmental conditions. In this state:

- Temperature remains within normal limits.
- Carbon monoxide concentration remains low.
- Carbon dioxide concentration remains below critical thresholds.
- Soil moisture remains sufficient.

No emergency actions are required, and the system continues routine monitoring operations. The collected measurements are stored and displayed for observation purposes only.

4.13.4 Level 1 – LOW WARNING

The LOW WARNING level indicates the presence of minor environmental abnormalities that may contribute to future fire development.

Typical conditions include:

- Light smoldering gas detection.
- Reduced soil moisture levels.
- Elevated temperature readings.
- Increased carbon dioxide concentration.

- Although immediate danger is not present, continued monitoring is recommended to observe whether environmental conditions continue to deteriorate.

4.13.5 Level 2 – MEDIUM DANGER

The MEDIUM DANGER level indicates a significant increase in fire probability. This level occurs when multiple environmental indicators simultaneously suggest hazardous conditions.

Examples include:

- Light smoldering combined with elevated temperature.
- Light smoldering combined with critical soil dryness.
- Increased CO_2 concentration combined with low soil moisture.

At this stage, the system begins generating stronger warning notifications because environmental conditions may rapidly evolve into a fire event.

4.13.6 Level 3 – HIGH DANGER

The HIGH DANGER classification represents severe environmental conditions associated with active combustion or a very high probability of fire occurrence.

Typical trigger conditions include:

- Heavy smoldering combustion.
- Heavy smoldering combined with dry soil conditions.
- Elevated carbon dioxide concentration combined with high temperature.

The detection of these conditions indicates that immediate attention is required and emergency personnel should be alerted.

4.13.7 Level 4 – CRITICAL

The CRITICAL level represents the highest danger category implemented in the proposed system.

This level is triggered when:

- Carbon dioxide concentration exceeds 2000 ppm.
- Air temperature exceeds 60°C.
- Soil temperature exceeds 60°C.
- Heavy smoldering is accompanied by elevated temperature conditions.

These conditions strongly indicate active fire development or an imminent fire event requiring immediate emergency response. Once a CRITICAL condition is detected, the system automatically activates all notification mechanisms and updates the cloud monitoring platform.

4.13.8 Notification Response According to Danger Level

Table 4.10: ***Summarizes the actions taken by the system for each danger level.***

Danger Level	***Firebase Update***	***Email Alert***	***Voice Call***
NORMAL	*Yes*	*No*	*No*
LOW WARNING	*Yes*	*Yes*	*No*
MEDIUM DANGER	*Yes*	*Yes*	*Yes*
HIGH DANGER	*Yes*	*Yes*	*Yes*
CRITICAL	*Yes*	*Yes*	*Yes*

4.13.9 Benefits of Multi-Level Classification

The implemented classification strategy offers several advantages:

- Early fire detection.
- Reduced false alarms.
- Better risk assessment accuracy.
- Progressive alert escalation.
- Improved emergency response planning.
- Enhanced situational awareness for system operators.

By evaluating multiple environmental indicators simultaneously and assigning graduated danger levels, the proposed system achieves more reliable fire detection compared with conventional single-threshold monitoring systems.

4.14 Cloud Monitoring and Notification System

4.14.1 Introduction

Real-time monitoring and rapid notification are essential requirements for any forest fire detection system. Detecting hazardous environmental conditions alone is insufficient unless the information can be delivered quickly to responsible personnel. Therefore, the proposed system integrates cloud monitoring services and automated notification mechanisms to ensure continuous accessibility to sensor data and immediate dissemination of emergency alerts.

The notification subsystem combines Firebase Realtime Database, e-mail notifications, and automated voice calls to provide multiple communication channels for emergency response.

4.14.2 TCP Communication between LabVIEW and Python

Communication between LabVIEW and the Python decision-making module is established through a local TCP connection. LabVIEW acts as the client, while the Python application operates as a TCP server listening on port 5000.

After sensor acquisition and signal conditioning, LabVIEW transmits a formatted comma-separated data string containing:

- Soil Moisture
- CO_2 Concentration
- CO Concentration
- Soil Temperature
- Air Temperature
- Node 1 Link Quality
- Node 2 Link Quality

The Python module receives the transmitted data, executes the fire detection algorithm, and returns the calculated danger level to LabVIEW. This communication mechanism enables real-time integration between data acquisition and intelligent decision-making processes.

4.14.3 Firebase Realtime Database

Firebase Realtime Database serves as the cloud storage and remote monitoring platform of the proposed system. Following each processing cycle, the latest sensor measurements and classification results are uploaded to the database.

The uploaded information includes:

- Soil Moisture
- Carbon Dioxide Concentration
- Carbon Monoxide Concentration
- Soil Temperature
- Air Temperature
- Node Link Quality Values
- System Status
- Danger Level
- Classification Description

This cloud-based architecture enables remote access to real-time environmental conditions from any internet-connected device. It also provides a centralized location for storing the most recent system status and alarm information.

4.14.4 E-Mail Notification System

The proposed system automatically generates e-mail notifications whenever abnormal environmental conditions are detected.

When the danger level reaches **LOW WARNING (Level 1)** or higher, an alert e-mail is generated and transmitted to predefined recipients. Each message contains:

- Current danger status.
- Fire risk description.
- Sensor measurements.
- Timestamp information.

The e-mail notification mechanism is implemented using Python's SMTP library and Gmail SMTP services. A cooldown timer is incorporated to prevent repetitive alerts during prolonged fire conditions.

4.14.5 Automated Voice Call System

To ensure immediate awareness during critical situations, the proposed system employs automated voice calls through the Twilio communication platform.

Whenever the danger level reaches MEDIUM DANGER (Level 2) or higher, the system automatically initiates phone calls to predefined emergency contacts. A synthesized voice message communicates:

- Current danger level.
- Fire status.
- Environmental condition description.

This mechanism provides an additional layer of reliability because phone calls can attract attention even when e-mail notifications are overlooked.

4.14.6 Notification Strategy

The notification strategy follows a progressive escalation approach.

*Table 4.11: **Notification Escalation Strategy***

Danger Level	***Firebase***	***Email***	***Voice Call***
Normal	✓	✗	✗
Low Warning	✓	✓	✗
Medium Danger	✓	✓	✓
High Danger	✓	✓	✓
Critical	✓	✓	✓

This approach minimizes unnecessary interruptions while ensuring that severe conditions receive immediate attention from responsible personnel.

4.14.7 Advantages of Cloud-Based Monitoring

The integration of cloud services and automated notification mechanisms provides several advantages:

- Real-time remote monitoring.
- Continuous data accessibility.
- Rapid emergency notification.
- Multi-channel communication.
- Improved situational awareness.
- Enhanced emergency response capability.

- Increased system reliability.

By combining cloud connectivity with intelligent alert generation, the proposed system extends beyond simple environmental monitoring and becomes a complete early warning platform for forest fire prevention and risk management.

4.15 Expected Outcomes

The proposed system is expected to provide continuous monitoring of environmental conditions and identify fire-related hazards at an early stage. Through the integration of wireless sensor networks, intelligent data processing, cloud monitoring, and automated notifications, the system can significantly improve fire detection efficiency and reduce response time.

The expected outcomes include:

- Real-time environmental monitoring.
- Early detection of fire indicators.
- Multi-level danger classification.
- Cloud-based data accessibility.
- Automated emergency notifications.
- Improved reliability of forest fire monitoring operations.

The successful implementation of these objectives contributes to reducing environmental losses and enhancing fire prevention strategies.

4.16 Chapter Summary

This chapter presented the problem formulation and system design of the proposed Wireless Sensor Network-based forest fire detection system. The chapter began with an analysis of the forest fire problem and the limitations of conventional monitoring approaches. System requirements, proposed architecture, and sensor selection criteria were subsequently discussed.

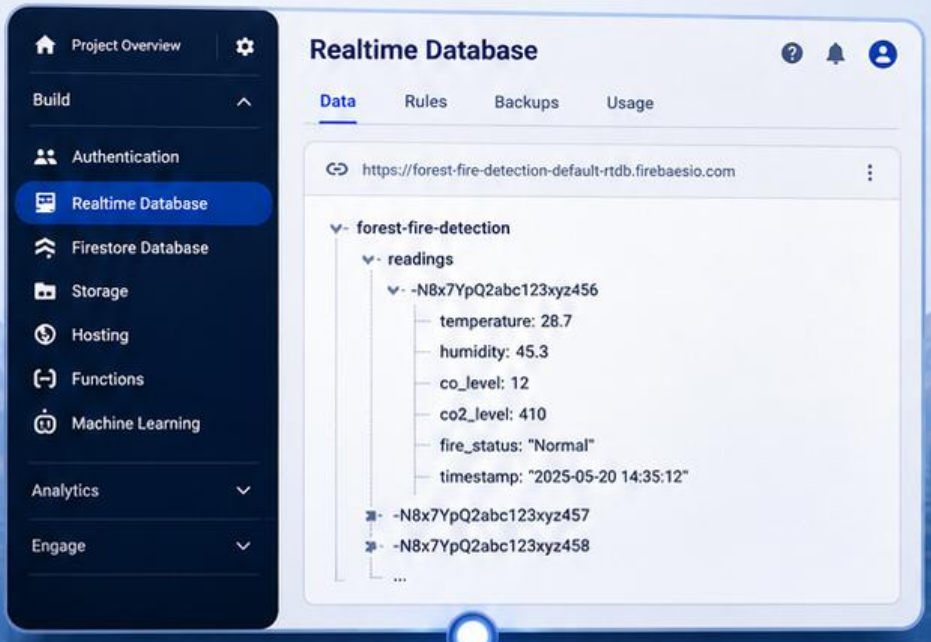
Detailed descriptions of the Soil Moisture Sensor, MQ-135 sensor, MQ-7 sensor, and K-Type Thermocouple were provided, including their operating principles, technical specifications, calibration procedures, and roles within the fire detection process. The chapter also introduced the Python-based fire detection algorithm and the multi-level danger classification strategy used for environmental risk assessment.

Finally, the cloud monitoring infrastructure, Firebase integration, e-mail notification system, and automated voice call mechanism were presented. Together, these components form a complete intelligent monitoring platform capable of providing early fire detection, real-time environmental monitoring, and rapid emergency notification.

CHAPTER

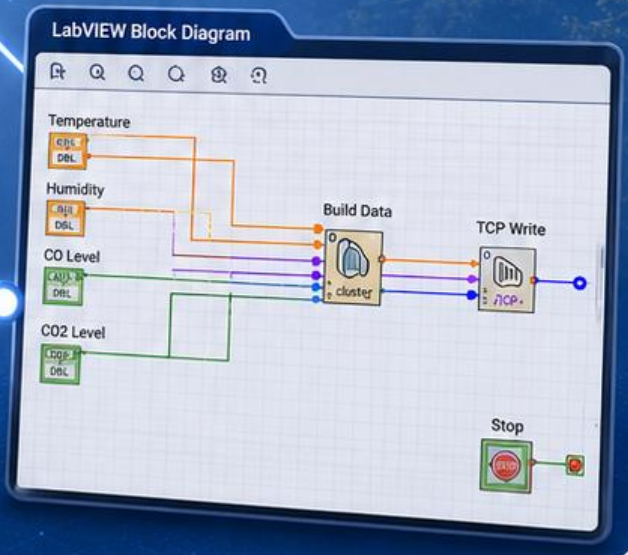
5

PROJECT IMPLEMENTATION



IoT Data Collection

Cloud Infrastructure



Real-time Data Ingestion | Python-based Processing | LabVIEW Integration | Firebase Realtime DB | Cloud Deployment | Secure & Scalable | Monitored & Visualized

5.1 Introduction

This chapter presents the complete software implementation of the Early Forest Fire Detection and Prediction System. The implementation encompasses two interdependent software layers: a LabVIEW-based data acquisition and control layer executing on the host PC, and a Python-based intelligent processing and communication backend. Together, these layers form a cohesive, real-time monitoring pipeline that collects environmental sensor data via NI WSN wireless sensor nodes, processes and classifies the data into danger levels, and disseminates alerts through multiple communication channels.

Figure 5.1: ***Full system integration.***

5.2 Preliminary Architecture — Common Sampling Rate Design

The preliminary architecture represents the initial prototype implementation of the forest fire detection system. In this design, all sensor acquisition loops operate under a single, unified sampling rate. This approach was adopted during the early development phase to simplify synchronization, reduce implementation complexity, and establish a functional baseline for integration testing between the LabVIEW front-end and the Python processing backend.

The preliminary architecture consists of four parallel sensor loops operating alongside one Main Control Loop. All loops are launched concurrently within a single LabVIEW VI using LabVIEW's inherent parallel execution model. A Queue-based inter-process communication mechanism ensures that sensor data is transferred to the Main Control Loop in a FIFO (First In, First Out) manner, preserving temporal ordering and preventing data loss during high-throughput periods.

5.2.1 System Architecture Overview

The preliminary system comprises five concurrent execution contexts within LabVIEW:

- Soil Moisture Acquisition Loop — reads analog voltage from the capacitive soil moisture sensor connected to AI0 of the NI WSN-3202 node.
- CO_2 Acquisition Loop — reads the MQ-135 gas sensor output from AI1 of the NI WSN-3202 node and converts the voltage to CO_2 concentration in parts per million (ppm).
- CO Acquisition Loop — reads the MQ-7 carbon monoxide sensor output from AI2 of the NI WSN-3202 node.
- Temperature Acquisition Loop — reads thermocouple channels TC0 (soil temperature) and TC1 (air temperature) from the NI WSN-3212 node.
- Main Control Loop — aggregates dequeued sensor data, formats and transmits a TCP packet to the Python server, receives the classified danger level response, updates visual indicators, and adjusts sampling parameters.

5.2.2 Queue Communication Architecture

The preliminary architecture employs LabVIEW's Queue primitives to implement a Producer-Consumer design pattern. Each sensor loop acts as a Producer, enqueuing calibrated sensor readings into a dedicated typed queue. The Main Control Loop acts as the Consumer, dequeuing data from all queues before assembling the TCP transmission packet.

Queue Lifecycle

Table 5.1: LabVIEW Queue Operation Summary.

Queue Primitive	*LabVIEW Function*	*Description*
Obtain Queue	*Obtain Queue*	*Creates or references a typed queue with a specified name and data type. Returns a queue reference used by all subsequent operations.*
Enqueue Element	*Enqueue Element*	*Appends a data element to the rear of the queue (FIFO). Called by the sensor loop (Producer) after each calibrated reading.*
Dequeue Element	*Dequeue Element*	*Removes and returns the front element of the queue. Called by the Main Control Loop (Consumer). Blocks until data is available.*
Release Queue	*Release Queue*	*Destroys the queue and frees associated memory. Called once during VI shutdown to prevent memory leaks.*

The FIFO mechanism guarantees that the Main Control Loop always processes sensor readings in chronological order. If a sensor loop produces readings faster than the Main Loop consumes them, the queue buffers the excess data without data loss, providing inherent flow control. This architecture is particularly robust for the preliminary prototype, where all sensors share the same sampling interval.

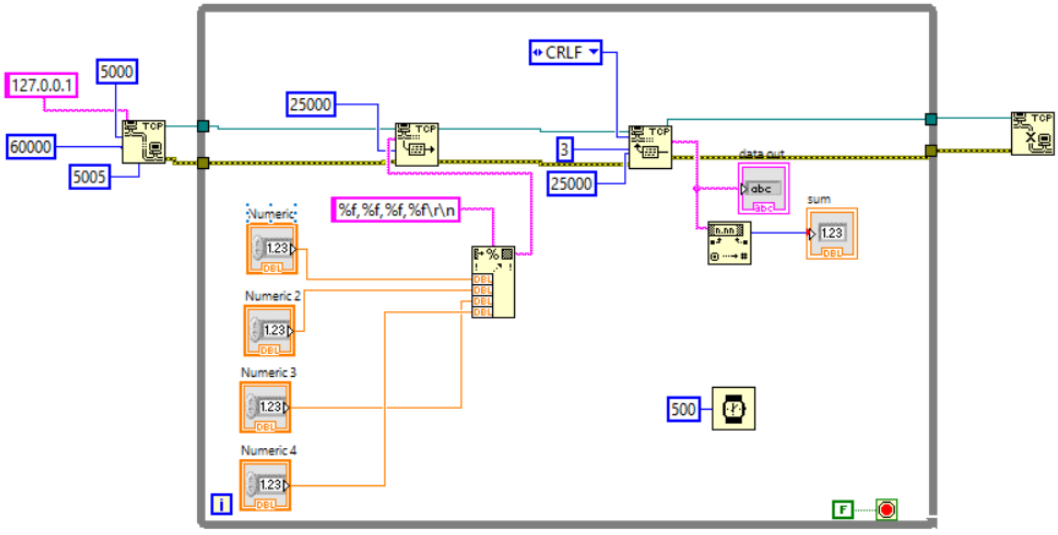


Figure 5.2: Experimental Verification of TCP Communication between LabVIEW and Python.

5.2.3 Common Sampling Rate Implementation

In the preliminary design, a single configurable timing constant governs all sensor loops simultaneously. The Wait (ms) primitive in each While Loop is wired to the same front panel control, ensuring that all sensors are sampled at the same rate. While this simplifies the implementation, it has the disadvantage that sensors with inherently different dynamics — for example, temperature changes slowly relative to CO concentration spikes — are not sampled at their optimal individual rates. This limitation motivated the development of the Enhanced Final Architecture described in Section 5.3.

*Table 5.2: **Preliminary Architecture — Common Sampling Rate***

Sensor Loop	***Measured Parameter***	***Node***	***Sampling Rate (Preliminary)***
Soil Moisture	*Soil Moisture %*	*WSN-3202 / AI0*	*Common Rate*
CO_2 (MQ-135)	*CO_2 ppm*	*WSN-3202 / AI1*	*Common Rate*
CO (MQ-7)	*CO ppm*	*WSN-3202 / AI2*	*Common Rate*
Temperature	*$T_{mo}{}^{IL}$ & $T_a{}^{Ir}$ °C*	*WSN-3212 / TC0,TC1*	*Common Rate*

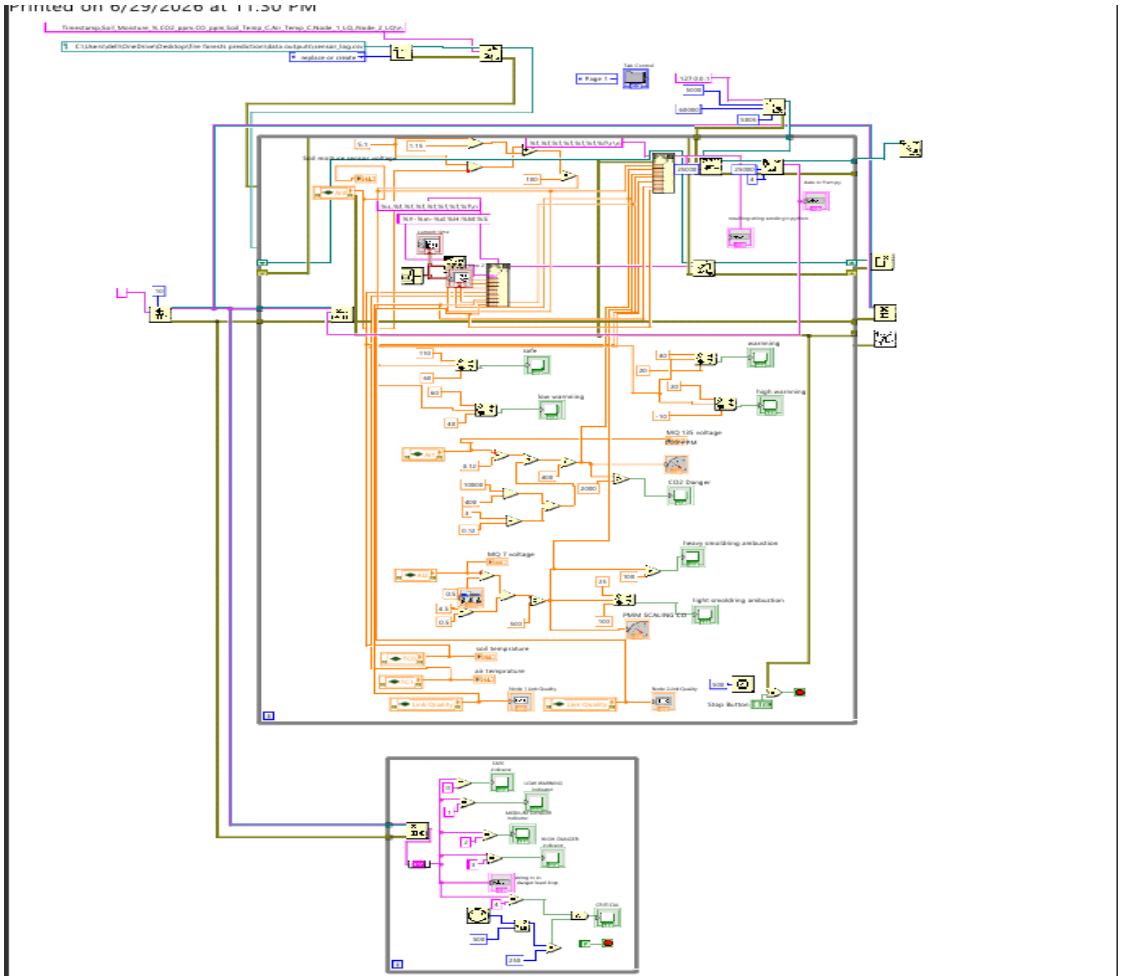


*Figure 5.3: **Full pellimantry archititure labview code.***

5.3 LabVIEW Virtual Instrument Implementation

The LabVIEW implementation consists of five Virtual Instruments (VIs) executing in parallel within the main project VI. Each sensor VI encapsulates the complete data acquisition, calibration, classification, CSV logging, and inter-process communication logic for its respective sensor. The Main Control Loop VI coordinates TCP communication with the Python backend and dynamically manages sensor sampling rates.

5.3.1 Soil Moisture Sensor VI

The Soil Moisture Sensor VI acquires the raw analog voltage from the capacitive soil moisture sensor connected to channel AI0 of the NI WSN-3202 wireless sensor node. The VI applies a two-point linear voltage-to-percentage conversion, classifies the resulting moisture percentage into one of four agronomic moisture categories, logs the timestamped reading to a CSV file, and enqueues the calibrated value for the Main Control Loop.

Voltage-to-Moisture Conversion

The sensor output voltage range is bounded by a dry reference voltage Vdry = 4.96 V and a wet reference voltage Vwet = 1.17 V. The linear conversion equation applied in the LabVIEW block diagram is:

Moisture (%) = [(4.96 - V_in) / (4.96 – 1.17)] * 100

Moisture Classification Thresholds

Table 5.3: Soil Moisture Classification Thresholds

Category	*ADC / Voltage Range*	*SM% Range*	*Indicator*
Saturated	*$V > V^{WR}_T$ (SM > 110%)*	*SM ≥ 110 %*	*Green*
Moist	*60 ≤ SM < 110 %*	*60–99 %*	*Blue*
Dry	*40 ≤ SM < 60 %*	*40–59 %*	*Yellow*
Very Dry	*SM < 40 %*	*< 40 %*	*Red*

A Very Dry classification (SM < 40%) is a critical pre-fire indicator, as dry organic matter has significantly reduced thermal capacity and lower ignition energy requirements. This threshold is propagated to the Python danger classification engine as a primary risk factor.

The VI logs each reading to soil_moisture_percentage.csv with the format [Timestamp, Soil_Moisture_%]. The timestamp is generated using LabVIEW's Get Date/Time String function with the format specifier %Y-%m-%d %H:%M:%S, ensuring ISO 8601 compliance for subsequent data analysis.

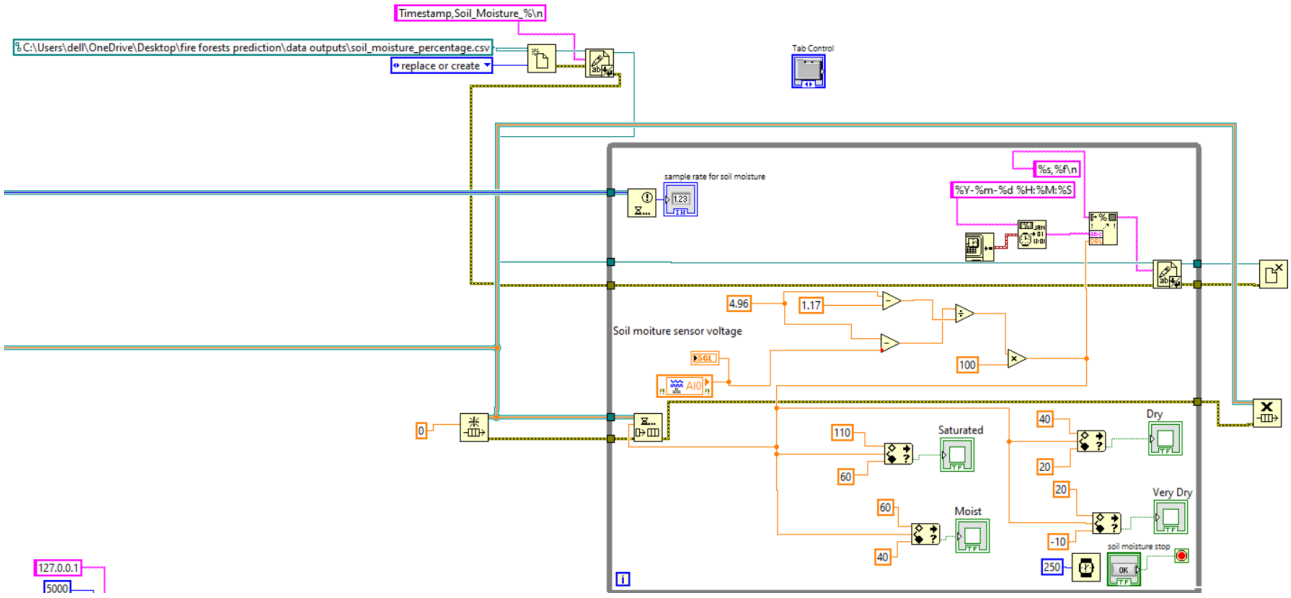


Figure 5.4: Soil Moisture Sensor VI — LabVIEW Block Diagram

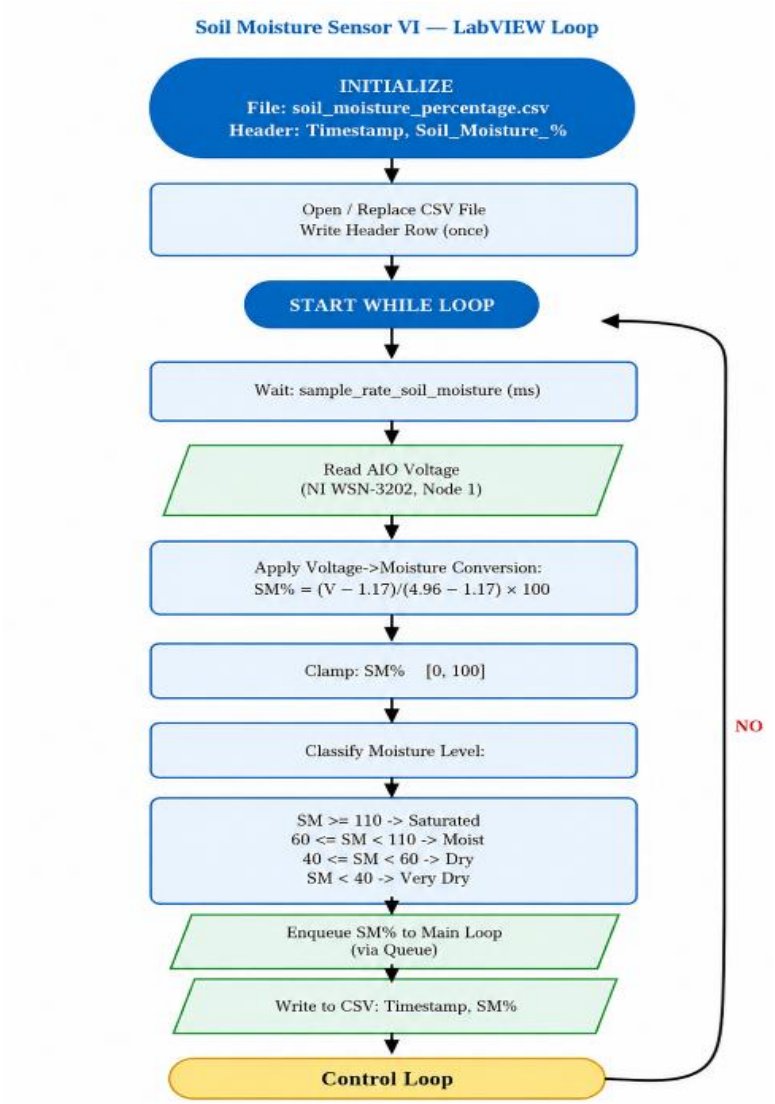


Figure 5.5: Soil Moisture Sensor VI — Processing Flowchart.

5.3.2 CO_2 Sensor VI (MQ-135)

The CO_2 Sensor VI acquires the analog voltage output from the MQ-135 gas sensor connected to channel AI1 of the NI WSN-3202 wireless sensor node. The MQ-135 is a metal-oxide semiconductor (MOS) gas sensor whose resistance varies non-linearly with the ambient CO_2 concentration. A two-point linear calibration procedure, referenced to the sensor's known response at 400 ppm (atmospheric baseline) and 10,000 ppm, is implemented directly in the LabVIEW block diagram.

Two-Point Linear Calibration Equation

Let V_0 be the sensor output voltage at C_0 = 400 ppm (atmospheric CO_2 baseline) and V_1 be the sensor output voltage at C_1 = 10,000 ppm (high concentration calibration point). The slope m and intercept b of the linear calibration model are:

$$m = (C_1 - C_0) / (V_1 - V_0)$$

$$b = C_0 - m \times V_0$$

$$CO_2 \text{ (ppm)} = m \times V_{mEas} + b$$

The LabVIEW block diagram implements this equation using the following numeric constants extracted from the calibration procedure: scaling factor 0.25, offset terms 400 and 10,000 ppm, and a gain of 4.57. The calibration constants are embedded as hardwired numeric controls within the block diagram to ensure repeatability across system restarts.

CO_2 Danger Threshold

A CO_2 concentration exceeding 2,000 ppm is classified as a fire-indicating condition. This threshold is based on the observed CO_2 elevation caused by smoldering combustion and early-stage pyrolysis in forest environments, where CO_2 levels can reach 2,000–5,000 ppm within the fire perimeter. When this threshold is exceeded, the CO_2 Danger boolean indicator on the front panel transitions to TRUE, and the danger classification in Python immediately assigns CRITICAL status (Level 4).

The VI logs each reading to co2_ppm.csv with the format [Timestamp, CO2_ppm].

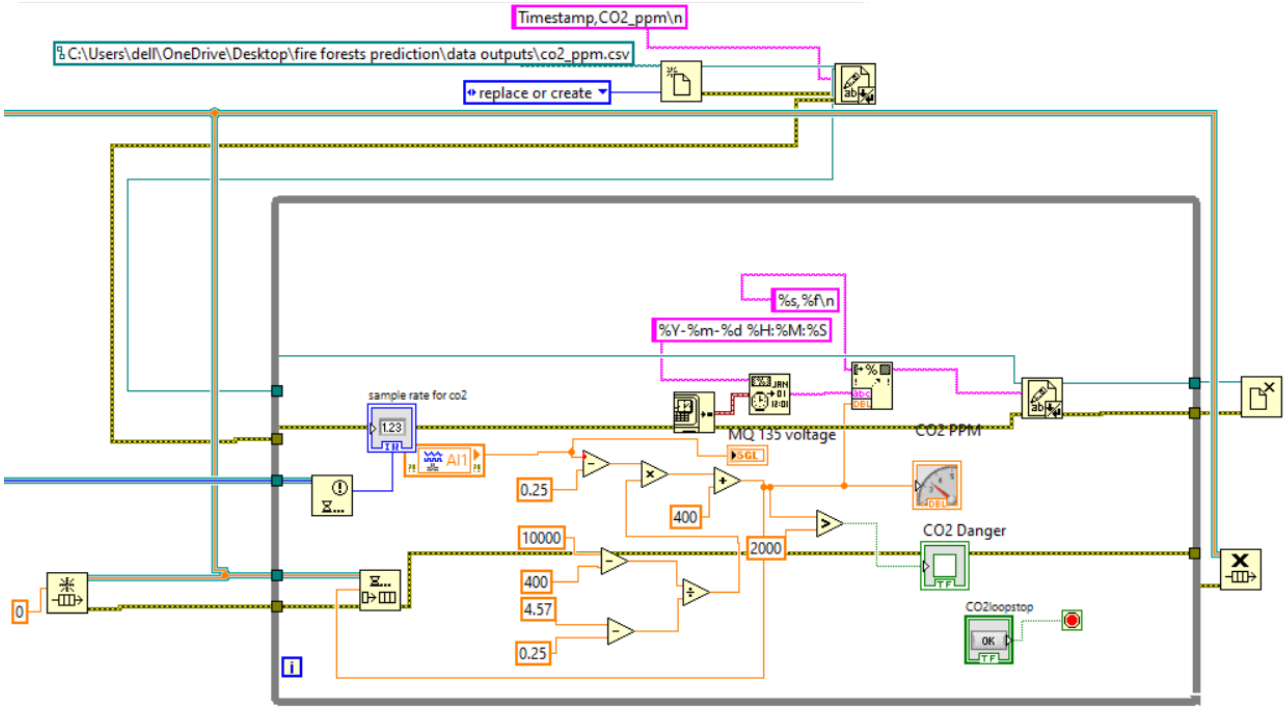


Figure 5.6: CO_2 Sensor VI (MQ-135) — LabVIEW Block Diagram

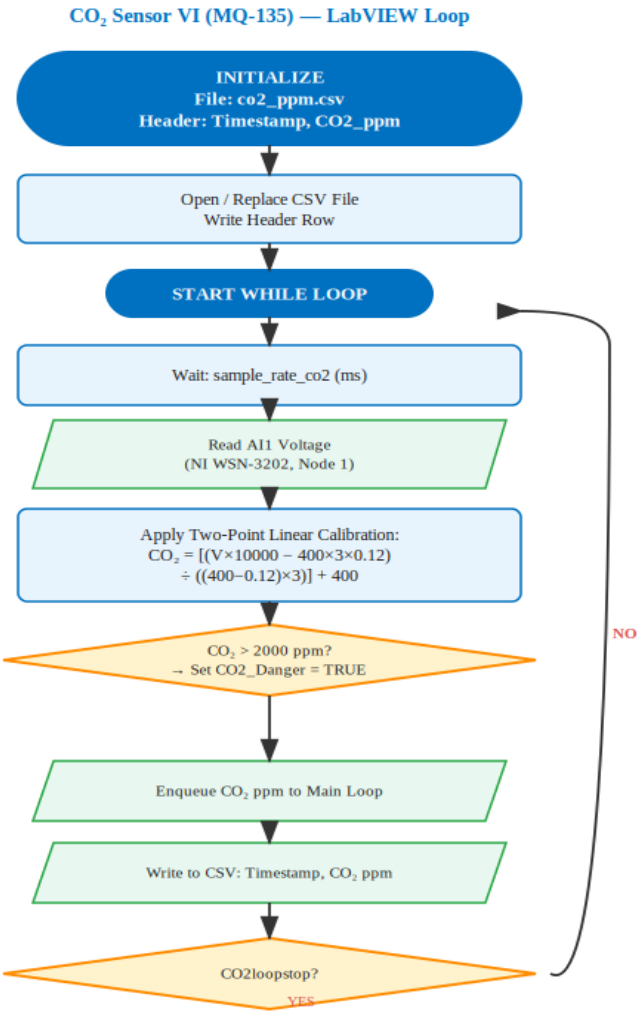


Figure 5.7: CO_2 Sensor VI — Processing Flowchart

5.3.3 CO Sensor VI (MQ-7)

The CO Sensor VI acquires the output voltage from the MQ-7 carbon monoxide sensor connected to channel AI2 of the NI WSN-3202 node. The MQ-7 sensor operates on a cyclic heating principle: a high-temperature heating phase (5 V, 60 s) followed by a low-temperature measurement phase (1.4 V, 90 s). Within the measurement phase, the sensor resistance decreases in proportion to ambient CO concentration, producing a measurable voltage change at the analog output.

PMM Scaling — Voltage to CO ppm Conversion

The piece-wise monotonic mapping (PMM) approach is applied to scale the MQ-7 output voltage to CO concentration in parts per million. The scaling parameters are derived from the sensor's datasheet response curve at known CO concentrations. The conversion equation implemented in the LabVIEW block diagram is:

$$CO\ (ppm) = [(V_{mEas} - Vmin) / (Vmax - Vmin)] \times Cmaxscale$$

$$CO\ (ppm) = [(V_{mEas} - 0.48) / (2.32 - 0.48)] \times 500$$

where Vmin = 0.5 V is the minimum output voltage at zero CO concentration, Vmax = 4.5 V is the maximum output voltage at full-scale CO concentration, and Cmaxscale = 500 ppm is the full-scale CO concentration. The divisor 500 (LabVIEW constant) and voltage bounds are configured as hardwired numeric constants.

CO Classification and Combustion Indicators

Table 5.4: CO Concentration Classification Thresholds

Condition	*CO Range*	*LabVIEW Indicator*	*Danger Contribution*
No Combustion	*0–25 ppm*	*None*	*None*
Light Smoldering	*25–100 ppm*	*light_smoldering = TRUE*	*Level 1+*
Heavy Smoldering	*> 100 ppm*	*heavy_smoldering = TRUE*	*Level 3+*

Additionally, the CO loop acquires the Link Quality (LQ) metric from the NI WSN-3202 Node 1. The LQ value, ranging from 0 to 100, represents the wireless signal quality between the sensor node and the NI WSN-9791 gateway. This metric is logged alongside the CO measurement to the file CO_ppm&LQ_3202.csv for post-hoc communication quality analysis.

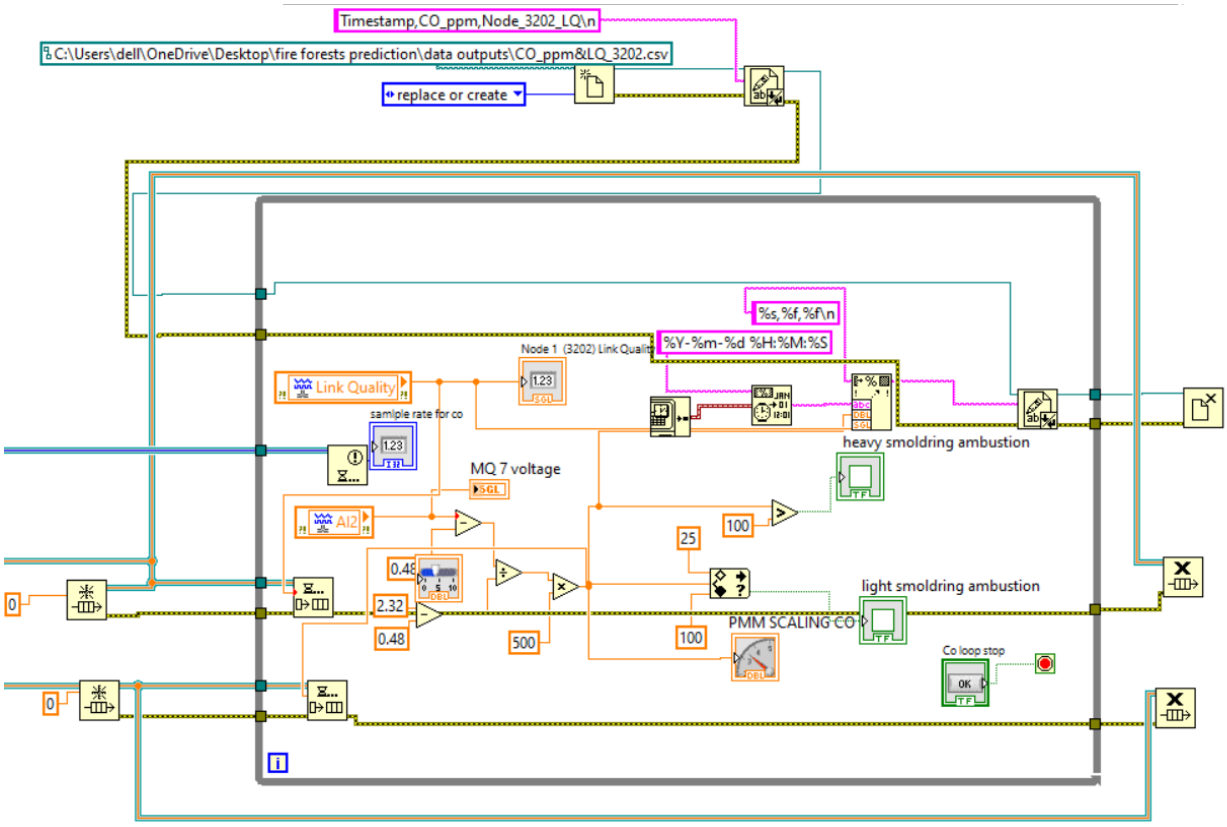


Figure 5.8: CO Sensor VI (MQ-7) — LabVIEW Block Diagram

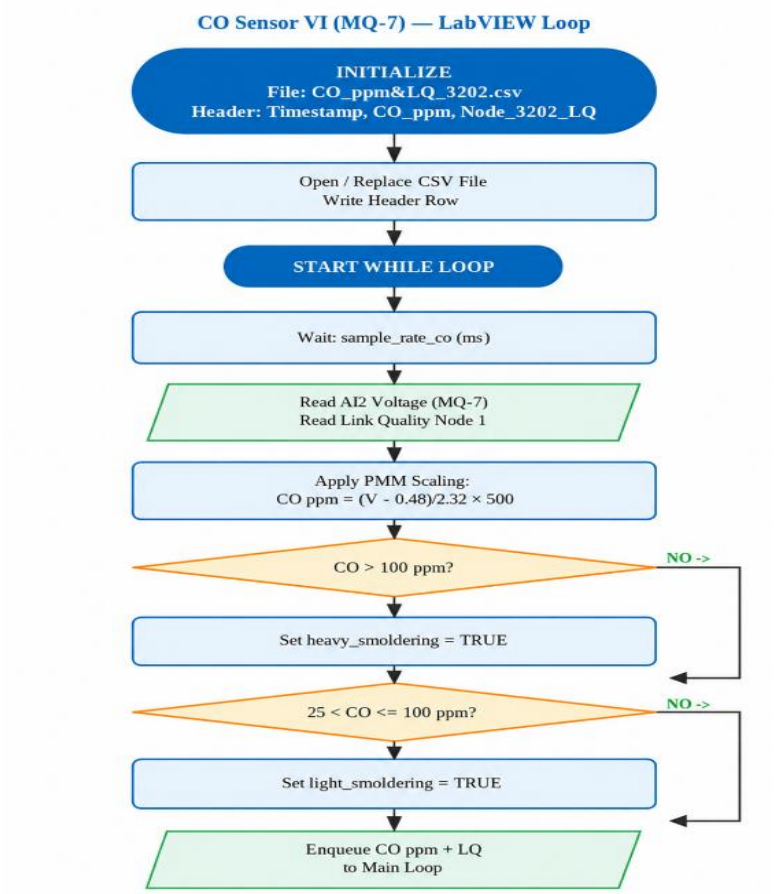


Figure 5.9: CO Sensor VI — Processing Flowchart

5.3.4 Temperature Sensor VI (Dual Thermocouple)

The Temperature Sensor VI acquires simultaneous soil and air temperature readings from the NI WSN-3212 wireless thermocouple input node. The WSN-3212 provides two thermocouple channels — TC0 and TC1 — with built-in cold-junction compensation (CJC) and linearization, returning calibrated temperatures directly in degrees Celsius without requiring additional software-side conversion.

Channel Assignments

Table 5.5: Temperature Sensor Channel Configuration

Channel	Sensor Type	Measured Parameter	Fire Threshold
TC0	K-Type Thermocouple	Soil Temperature (°C)	Soil T > 60°C → CRITICAL
TC1	K-Type Thermocouple	Air Temperature (°C)	Air T > 60°C → CRITICAL Air T > 40°C → HIGH
LQ	WSN Link Quality	Node 2 Link Quality (%)	LQ < 20% → Node Timeout

The VI records a Get Date/Time String timestamp at each acquisition cycle and writes the four-field record [Timestamp, Soil_Temperature, Air_Temperature, Node_3212_LinkQuality] to the file temp_soil_air_Node_3202_LQ.csv. The CSV output facilitates offline data analysis, machine learning model training, and regulatory compliance documentation.

The temperature danger thresholds are derived from established forestry fire science research. Soil temperatures exceeding 60°C indicate active pyrolysis of organic soil matter, while air temperatures exceeding 40°C represent conditions consistent with pre-fire atmospheric heating in forest microenvironments.

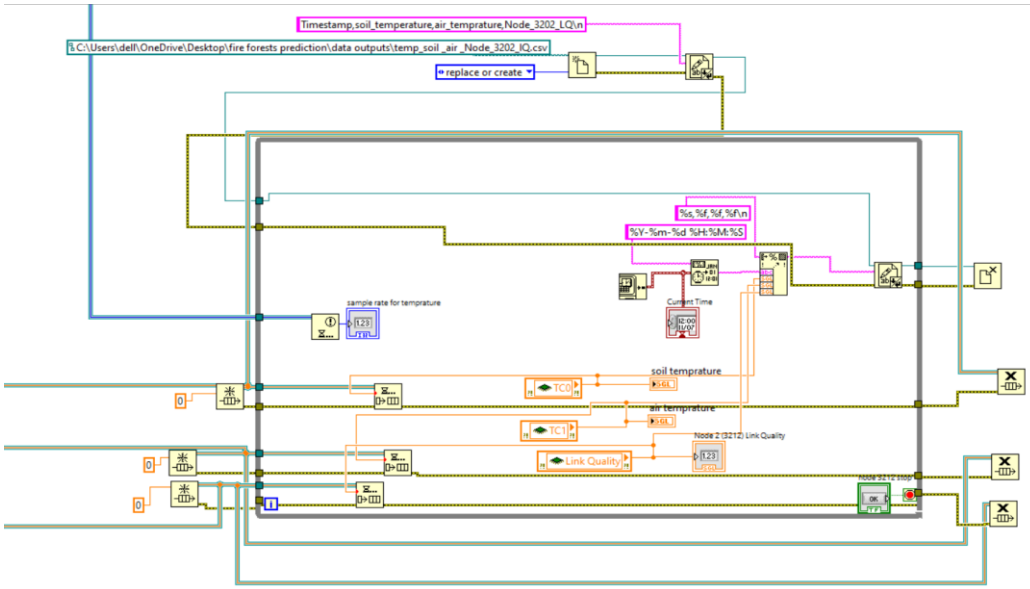


Figure 5.10: Temperature Sensor VI — LabVIEW Block Diagram

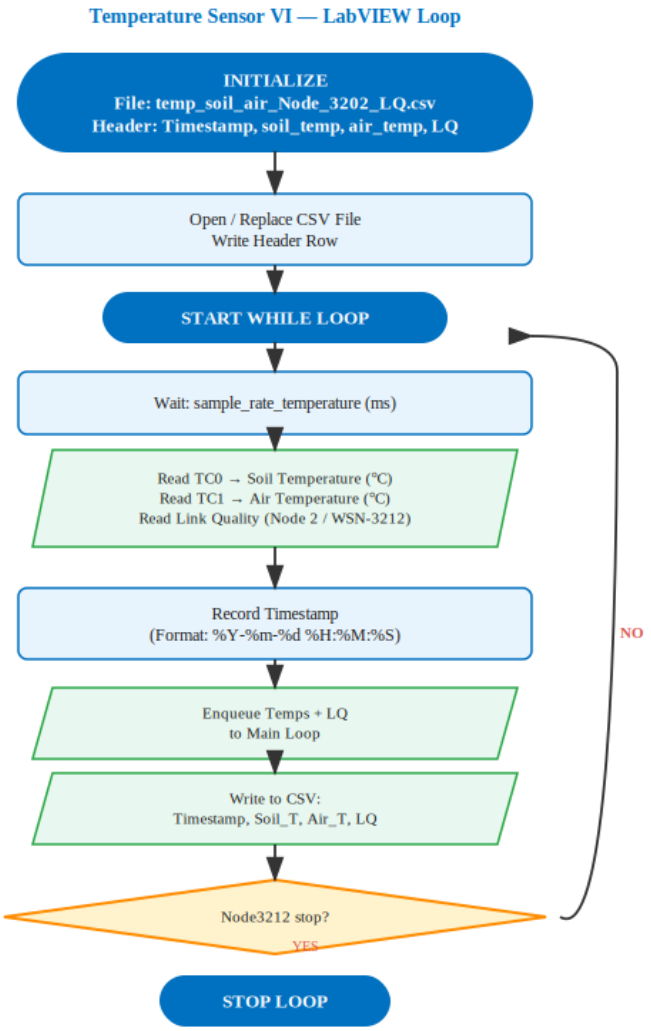


Figure 5.11: Temperature Sensor VI — Processing Flowchart

5.3.5 Main Control Loop VI

The Main Control Loop VI serves as the central coordinator of the LabVIEW implementation. It is responsible for TCP communication with the Python backend, aggregation of dequeued sensor data from all four sensor loops, danger level display, and dynamic control of sensor sampling rates. The loop executes at its own timing interval independent of the sensor loops, governed by a configurable wait constant of 400 ms by default.

TCP Communication Sequence

The Main Control Loop implements the complete TCP client-side communication sequence using LabVIEW's TCP/IP palette functions:

- TCP Open Connection: establishes a TCP connection to the Python server at address 127.0.0.1, port 5000, with a connection timeout of 60,000 ms and an open timeout of 5,005 ms. The connection is established once before the main While Loop begins execution.
- TCP Write: transmits a formatted ASCII string containing seven comma-separated sensor values followed by a carriage return and newline (\r\n) delimiter. The Format Into String function uses the format specifier %f,%f,%f,%f,%f,%f,%f\r\n.
- TCP Read: receives the Python server's response — the danger level integer (0–4) as an ASCII string — with a maximum byte count of 25,000 bytes and a CRLF termination character.
- TCP Close Connection: cleanly terminates the TCP session upon VI stop, ensuring proper socket cleanup and preventing port binding conflicts on subsequent VI launches.

TCP Packet Format

Table 5.6: TCP Transmission Packet — Field Definitions

Field	*Parameter*	*Source*	*Unit*	*Format*
1	*Soil Moisture*	*AI0 / WSN-3202*	*%*	*%f*
2	*CO_2 ppm*	*AI1 / WSN-3202*	*ppm*	*%f*
3	*CO ppm*	*AI2 / WSN-3202*	*ppm*	*%f*
4	*Soil Temperature*	*TC0 / WSN-3212*	*°C*	*%f*
5	*Air Temperature*	*TC1 / WSN-3212*	*°C*	*%f*
6	*Node 1 Link Q*	*WSN-3202 Gateway*	*%*	*%f*
7	*Node 2 Link Q*	*WSN-3212 Gateway*	*%*	*%f*

The response from Python is a single ASCII integer in the range 0–4 followed by \r\n. The String to Number function converts this response to a numeric danger level, which drives five front panel Boolean LED indicators: SAFE (0), LOW WARNING (1), MEDIUM DANGER (2), HIGH DANGER (3), and CRITICAL (4).

Dynamic Sampling Rate Control

A critical feature of the Main Control Loop is its ability to dynamically adjust the sampling rate of each sensor loop in response to the current danger classification. This mechanism is implemented via LabVIEW Notifiers, which broadcast updated sampling rate values to all four sensor loops whenever the danger level changes. The sampling rate lookup table is encoded as a Case Structure within the Main Control Loop:

Table 5.7: Dynamic Sampling Rate Table — Based on Danger Level

Danger Level	***Status***	***Soil Moisture***	***CO_2 (ms)***	***CO (ms)***	***Temperature (ms)***
0	*SAFE*	*6 hour*	*15 minute*	*15 minute*	*15 minute*
1	*LOW WARNING*	*3 hour*	*5 minute*	*5 minute*	*5 minute*
2	*MED DANGER*	*1 hour*	*1 minute*	*1 minute*	*1 minute*
3	*HIGH DANGER*	*15 minute*	*15 second*	*15 second*	*15 second*
4	*CRITICAL*	*5 minute*	*2 second*	*2 second*	*2 second*

Similarly, the temperature loop applies a secondary adaptive sampling scheme. At SAFE (Level 0), the temperature loop waits 360 ms per cycle; at LOW WARNING (Level 1), the wait is reduced to 180 ms; at higher danger levels (2–4), the temperature sampling interval drops to 60 ms, 15 ms, and 5 ms respectively. This ensures that critical thermal dynamics are captured with maximum temporal resolution when fire conditions are detected.

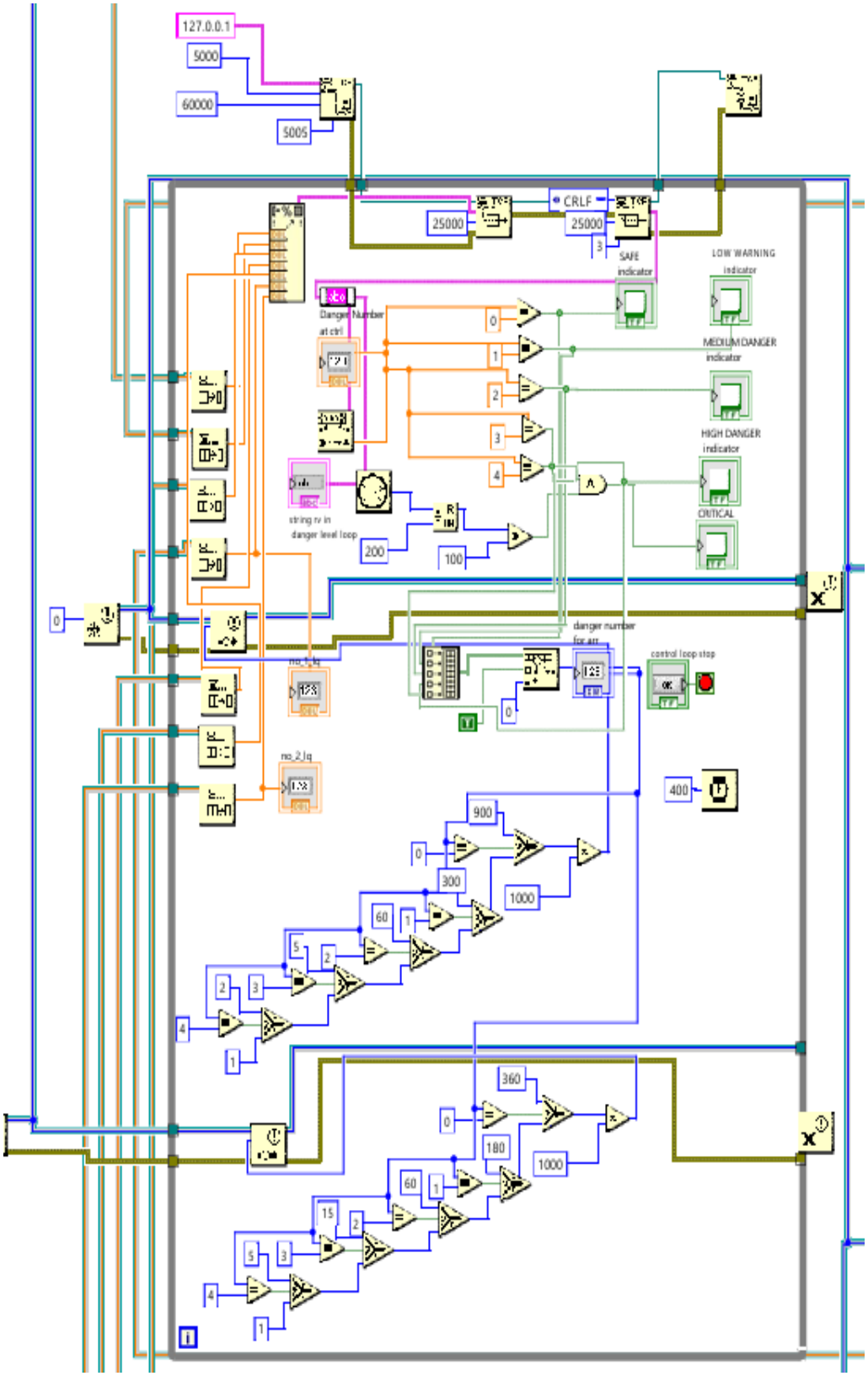


Figure 5.12: Main Control Loop VI — LabVIEW Block Diagram

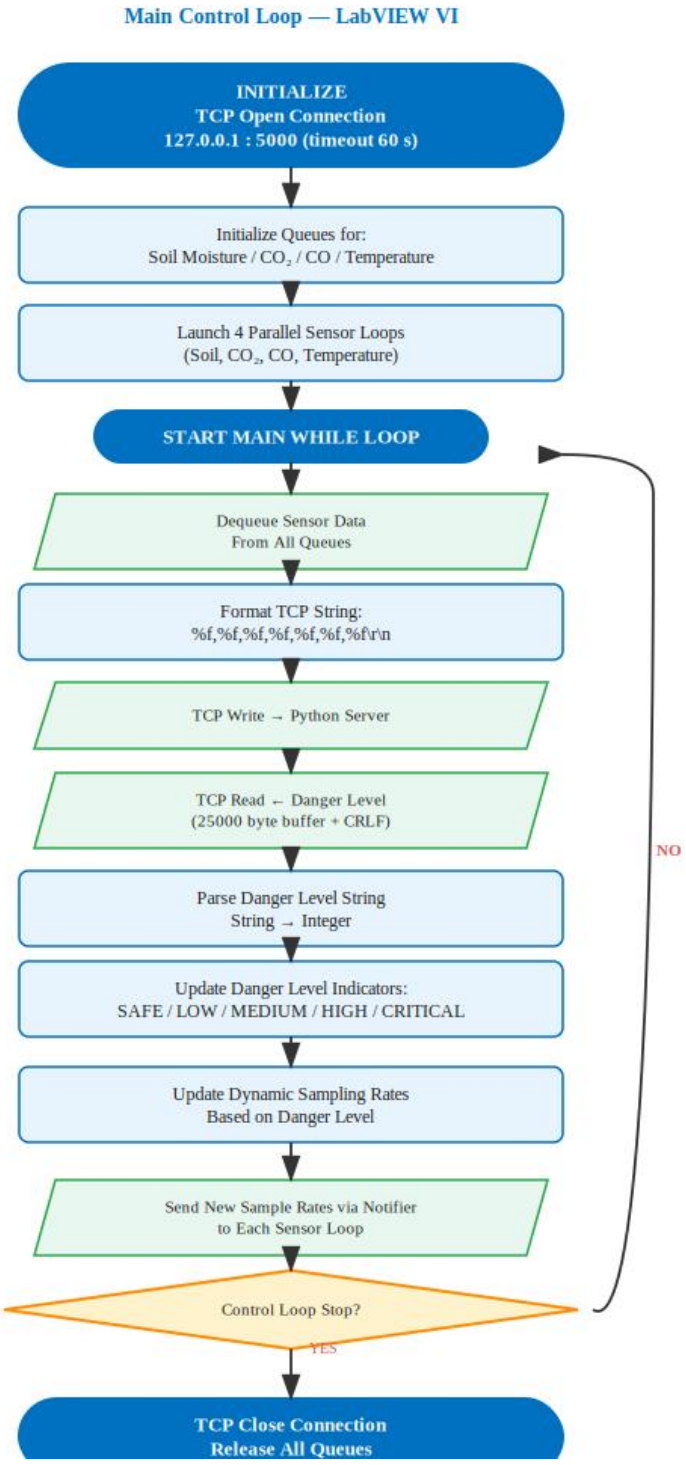


Figure 5.13: Main Control Loop — Processing Flowchart

5.4 TCP Communication — LabVIEW Client to Python Server

The TCP/IP communication layer forms the critical inter-process bridge between the LabVIEW data acquisition front-end and the Python intelligent processing backend. LabVIEW operates as the TCP client, initiating the connection and transmitting sensor data. Python operates as the TCP server, listening on the loopback interface and returning danger classification results.

5.4.1 Connection Parameters

Table 5.8: TCP Connection Configuration Parameters

Parameter	LabVIEW Setting	Python Setting
Server Address	127.0.0.1	HOST = '127.0.0.1'
Port Number	5000	PORT = 5000
Conn. Timeout	60,000 ms	N/A (server side)
Open Timeout	5,005 ms	server.listen(1)
Rx Buffer	25,000 bytes	conn.recv(1024)
Socket Timeout	N/A	conn.settimeout(20 s)
Terminator	CRLF (\r\n)	Decoded + stripped
Protocol	TCP SOCK_STREAM	AF_INET, SOCK_STREAM

5.4.2 Communication Sequence

The TCP communication follows a strict request-response cycle synchronized with the LabVIEW main While Loop timing:

- Step 1 — Server Bind: The Python script calls socket.bind((HOST, PORT)) and socket.listen(1), placing the server socket in the listening state on port 5000 of the loopback interface.
- Step 2 — Client Connect: The LabVIEW TCP Open Connection function attempts to establish a connection to 127.0.0.1:5000. Upon acceptance, conn.accept() returns a new socket object and the client address.
- Step 3 — Data Transmission: Each iteration of the LabVIEW main While Loop calls TCP Write with a formatted string of seven floating-point sensor values, comma-separated and terminated with \r\n.
- Step 4 — Server Processing: Python's conn.recv(1024) receives the packet, decodes it from bytes to a UTF-8 string, and invokes classify_sensors() to determine the danger level.
- Step 5 — Response: The Python server immediately sends back the danger level integer as an ASCII string followed by \r\n using conn.send((str(danger_level) + '\r\n').encode()).

- Step 6 — LabVIEW Parse: TCP Read receives the response string; the String to Number function extracts the numeric danger level for display and adaptive sampling rate updates.
- Step 7 — Timeout Handling: If no data is received within 20 seconds, Python logs a timeout error and continues waiting. LabVIEW's connection timeout prevents indefinite blocking.

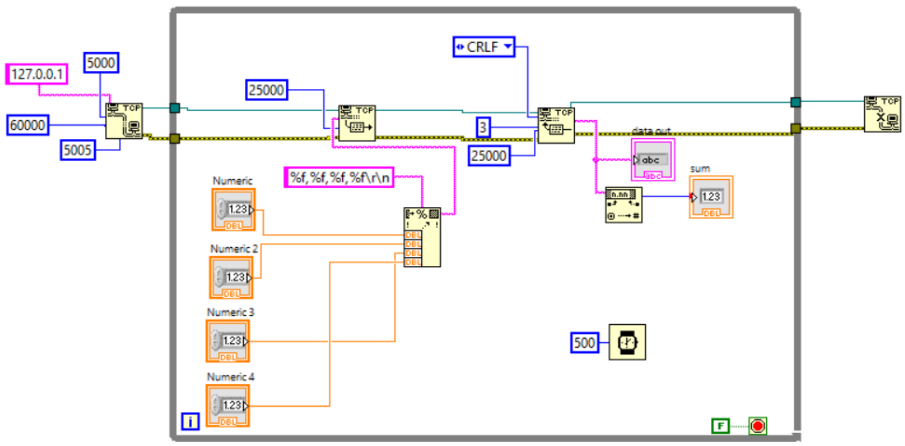


Figure 5.14: TCP Communication VI — LabVIEW Block Diagram

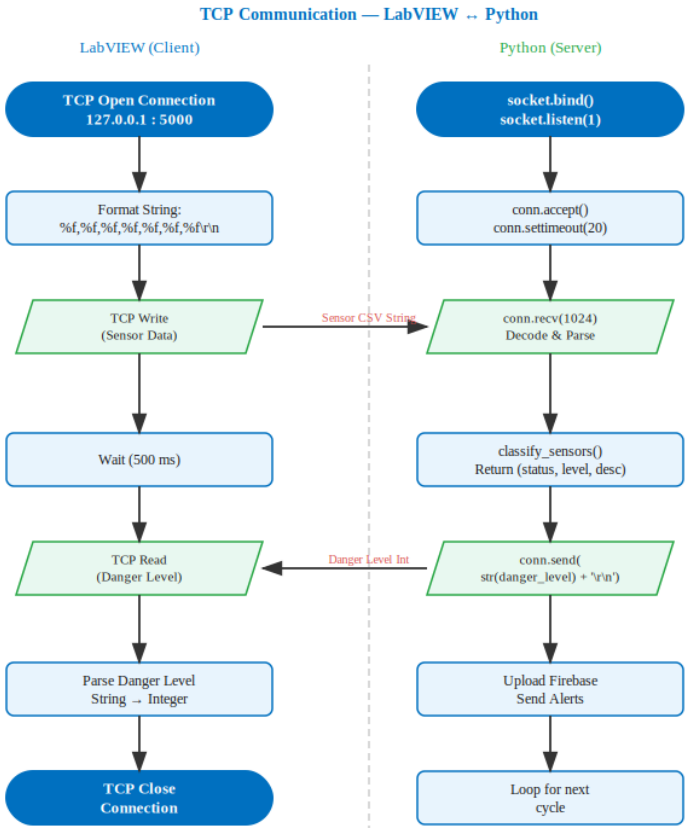


Figure 5.15: TCP Communication Sequence Flowchart

5.5 Python Backend Implementation

The Python backend serves as the intelligent processing engine of the forest fire detection system. It receives sensor data from LabVIEW via TCP, applies a multi-parameter danger classification algorithm, uploads the processed data to the Firebase Realtime Database, and dispatches automated alert notifications via email and voice call. The backend is implemented as a single-threaded synchronous application to ensure deterministic execution order and avoid race conditions in the alert dispatch logic.

5.5.1 Software Architecture and Module Dependencies

Table 5.9: ***Python Backend Module Dependencies***

Module	***Library/Package***	***Purpose***
socket	*Python stdlib*	*TCP/IP server socket for LabVIEW communication*
smtplib	*Python stdlib*	*SMTP email transmission via Gmail SSL*
firebase_admin	*firebase-admin*	*Firebase Realtime Database read/write operations*
twilio	*twilio*	*Voice call alerts to registered phone numbers*
math	*Python stdlib*	*NaN and Infinity detection (math.isnan, math.isinf)*
time	*Python stdlib*	*Alert cooldown timing via time.time()*

5.5.2 Sensor Validation and Sanitization

Before any classification or upload operation, the Python backend validates all incoming sensor readings for numerical validity. The is_bad() helper function returns True if a value is NaN (Not a Number), positive or negative infinity, or None — conditions that can arise from sensor disconnection, wiring faults, or LabVIEW floating-point conversion errors.

The sanitize_value() function replaces invalid readings with a caller-specified fallback: safe numeric defaults are used for classification purposes (preventing false alarms from invalid data), while the human-readable string 'No sensor / connectivity problem' is used for Firebase uploads to ensure the dashboard displays a meaningful error state rather than a null value.

Invalid sensor readings trigger a separate sensor error email notification with a 15-minute cooldown, identifying the specific sensors that returned invalid data and listing probable causes including physical disconnection, wiring faults, and LabVIEW signal conversion errors.

5.5.3 Danger Classification Algorithm — classify_sensors()

The classify_sensors() function implements a deterministic, rule-based multi-parameter danger classification algorithm that evaluates the combined state of all five sensor channels and returns a three-tuple: (status_string, danger_level_integer, description_string). The function applies a priority-ordered evaluation cascade, from the most severe condition (CRITICAL, Level 4) to the baseline condition (NORMAL, Level 0), ensuring that the highest applicable danger level is always returned.

Classification Boolean Flags

Table 5.10: Boolean Classification Flags — Definition and Thresholds

Flag Variable	*Condition*	*Physical Meaning*
heavy_smoldering	*CO > 100 ppm*	*Active combustion with heavy smoke production*
light_smoldering	*25 < CO ≤ 100 ppm*	*Incipient smoldering or early-stage combustion*
co2_danger	*CO_2 > 2000 ppm*	*Fire-level CO_2 indicating active pyrolysis*
co2_high	*500 < CO_2 ≤ 2000 ppm*	*Elevated CO_2, possible pre-fire off-gassing*
temp_critical	*Tair> 60°C or Tsoil > 60°C*	*Extreme heat consistent with fire proximity*
temp_high_warning	*40 < Tair ≤ 60°C*	*Abnormally high air temperature, fire possible*
moisture_critical	*SM < 40 %*	*Critically dry soil — high ignition susceptibility*

Classification Cascade — Priority Order

Table 5.11: Danger Classification Decision Table

Level	*Status*	*Condition(s)*	*Action*
4	*CRITICAL*	*co2_danger OR temp_critical OR (heavy_smoldering AND temp_high_warning)*	*Email + Call*
3	*HIGH DANGER*	*heavy_smoldering AND moisture_critical OR heavy_smoldering OR (co2_high AND temp_high_warning)*	*Email + Call*
2	*MEDIUM DANGER*	*light_smoldering AND (temp warning OR moisture_critical) OR (co2_high AND moisture_critical)*	*Email + Call*
1	*LOW WARNING*	*light_smoldering OR (moisture_critical AND temp_warning) OR (moisture_low AND co2_high) OR temp_high_warning*	*Email only*
0	*NORMAL*	*All readings within safe thresholds*	*None*

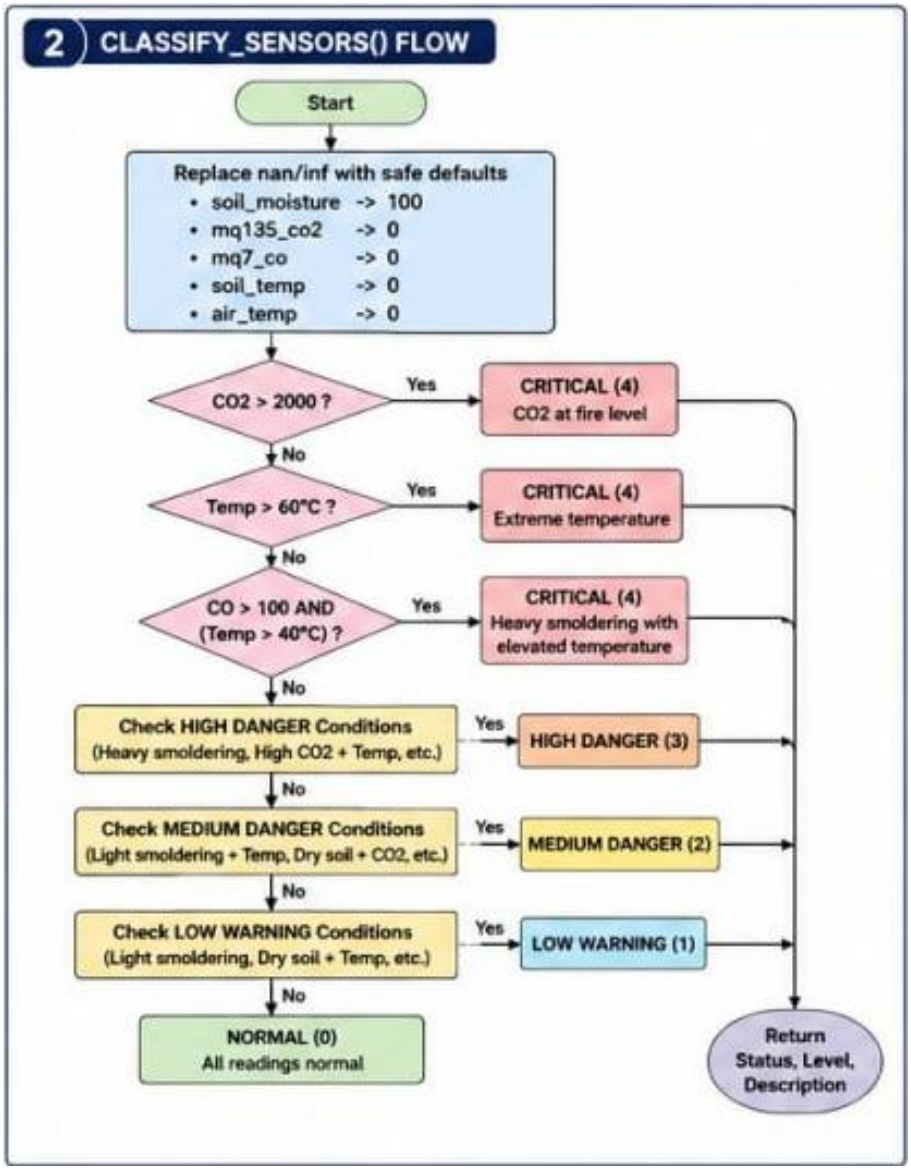


Figure 5.16: classify_sensors() — Danger Classification Flowchart

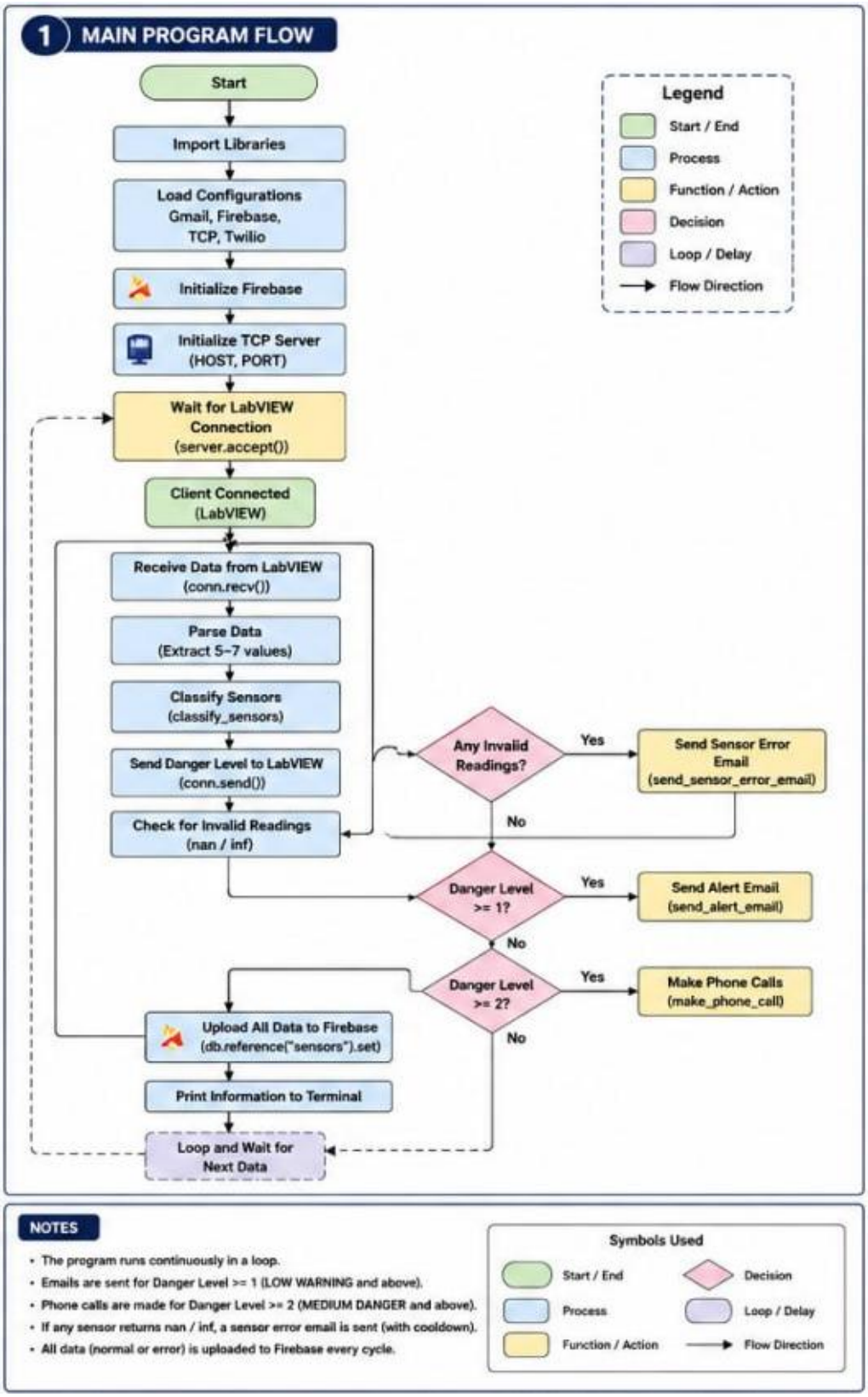


Figure 5.17: Python Main Program — Complete Execution Flowchart

5.5.4 Firebase Realtime Database Integration

The Firebase Realtime Database provides persistent, cloud-accessible storage for all processed sensor readings and the current system status. The Python backend uses the firebase-admin SDK to authenticate with the database using a service account credentials file, and performs atomic updates to the /sensors node on every processing cycle.

A single db.reference('sensors').set(payload) call replaces the complete sensor node contents on each cycle, ensuring that the dashboard always reflects the most recent system state without accumulating stale data. The payload dictionary contains ten fields:

Table 5.12: ***Firebase /sensors Node — Field Definitions***

Firebase Field	***Data Type***	***Content***
soil_moisture	*Float / String*	*Calibrated soil moisture % or error string*
mq135_co2	*Float / String*	*CO_2 concentration in ppm or error string*
mq7_co	*Float / String*	*CO concentration in ppm or error string*
soil_temperature	*Float / String*	*Soil temperature in °C or error string*
air_temperature	*Float / String*	*Air temperature in °C or error string*
node1_link_quality	*Float / String*	*WSN-3202 link quality % or error string*
node2_link_quality	*Float / String*	*WSN-3212 link quality % or error string*
status	*String*	*Danger status label (e.g., 'HIGH DANGER')*
danger_level	*Integer (0–4)*	*Numeric danger level for dashboard logic*
description	*String*	*Human-readable alert description*

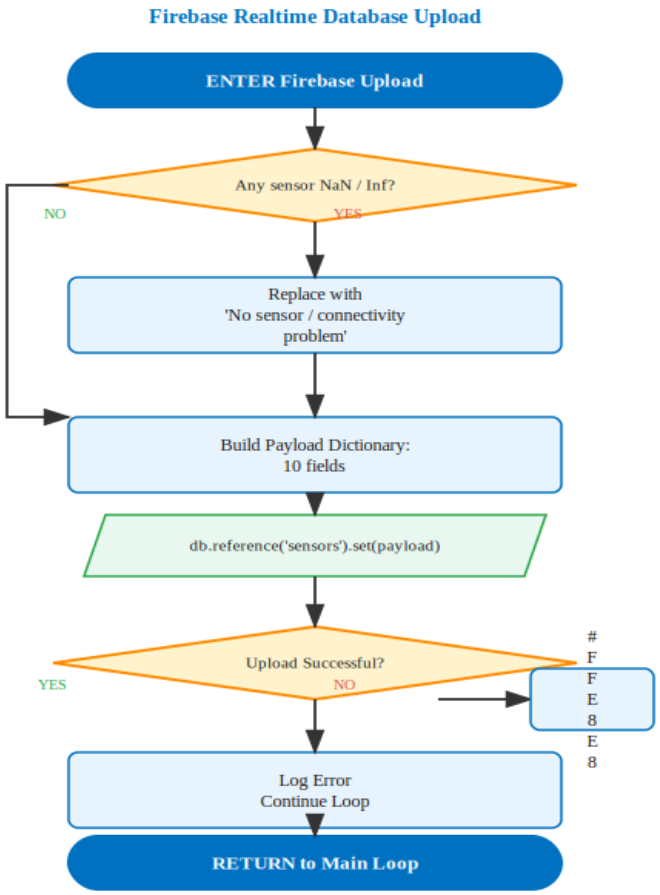


Figure 5.18: Firebase Upload — Processing Flowchart

5.5.5 Alert Notification System

The alert notification system implements three distinct communication channels, each with independent cooldown timers to prevent alert flooding during sustained fire events:

Email Alerts (Gmail SMTP)

Email alerts are dispatched for danger levels ≥ 1 (LOW WARNING and above) using Python's smtplib module with SSL encryption over port 465. The send_alert_email() function constructs a MIMEMultipart email message containing a formatted plaintext body with all current sensor readings, the danger status, and the descriptive alert message. The email is transmitted simultaneously to five recipient addresses using a single sendmail() call.

A cooldown timer of 5 minutes (300 seconds) prevents repeated emails during sustained danger conditions. The cooldown is implemented using time.time() comparisons against the _last_alert_email_time variable, which is updated on every successful email transmission.

Voice Call Alerts (Twilio)

Voice call alerts are dispatched for danger levels ≥ 2 (MEDIUM DANGER and above) using the Twilio REST API. The make_phone_call() function constructs a TwiML (Twilio Markup Language) XML response containing a <Say> verb with the danger status and description, spoken by the 'alice' voice in en-US locale. The call is initiated to all numbers in the CALL_RECIPIENTS list. A separate 5-minute cooldown timer (_last_phone_call_time) governs call frequency.

Sensor Error Alerts

A dedicated alert channel handles the case where one or more sensors return NaN or infinity values. The send_sensor_error_email() function dispatches an email identifying the specific failed sensors and listing probable causes. A 15-minute cooldown (NAN_EMAIL_COOLDOWN = 900 s) prevents repeated error notifications during extended hardware faults.

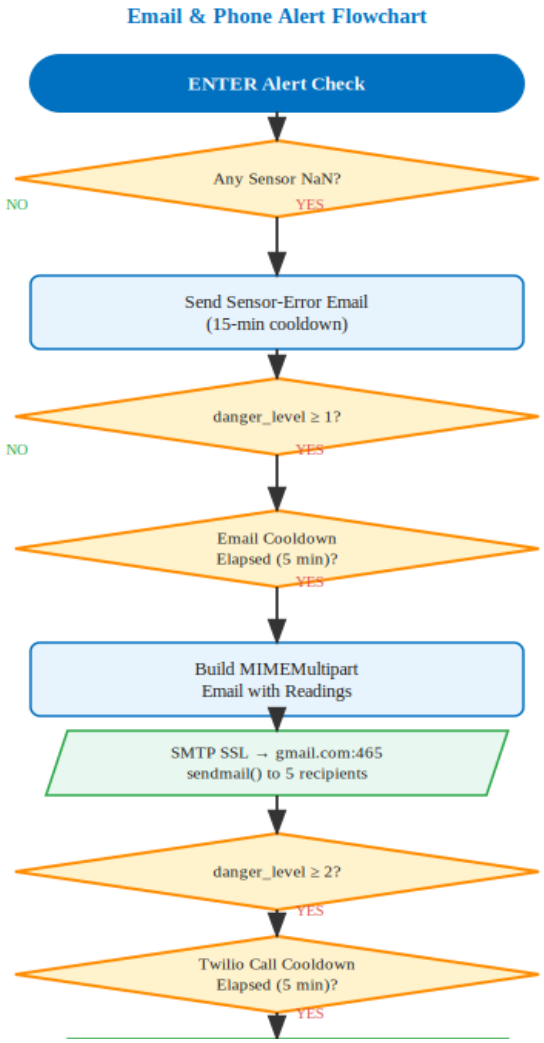


Figure 5.19: Email and Phone Alert System — Flowchart

5.6 Enhanced Final System Architecture

The Enhanced Final Architecture represents a significant optimization of the preliminary design, addressing the limitations identified during integration testing of the common-sampling-rate prototype. The enhanced implementation introduces independent sensor loops, each operating at a dynamically adjustable sampling rate determined by the current danger classification. A Notifier-based inter-process communication mechanism replaces direct variable sharing for control signal propagation, while the Queue-based data path is retained for sensor data aggregation.

5.6.1 Architectural Improvements Overview

The key improvements introduced in the Enhanced Final Architecture relative to the preliminary design are:

- Independent Sampling Rates: Each of the four sensor loops operates at its own configurable sampling rate, dynamically adjusted by the Main Control Loop via Notifiers. This enables the system to allocate CPU and radio communication resources proportionally to the risk level detected by each sensor channel.
- Event-Driven Sampling Rate Updates: Notifiers implement a non-blocking publish-subscribe mechanism whereby the Main Control Loop publishes updated sampling rates and each sensor loop receives the update asynchronously without requiring synchronization barriers.
- Reduced Power Consumption: By operating sensors at low sampling rates during SAFE conditions and only increasing rates when danger is detected, the enhanced architecture significantly reduces the duty cycle of the NI WSN sensor nodes, extending battery life.
- Improved CPU Utilization: Decoupled loop timing prevents the Main Control Loop from being blocked by sensor acquisition delays, improving system responsiveness and throughput.
- Better Scalability: Additional sensor types can be integrated by introducing new sensor loops without modifying the Main Control Loop's core logic, as the Queue and Notifier interfaces provide clean abstraction boundaries.

5.6.2 Notifier Architecture

LabVIEW Notifiers implement a publish-subscribe messaging pattern that is distinctly different from the Queue mechanism used for sensor data. While a Queue enforces FIFO

ordering and blocks the consumer until data is available, a Notifier is a lightweight, non-buffered event signal: the most recent value overwrites any unread previous value, and a waiting loop receives the update immediately. This behavior makes Notifiers ideally suited for control messages such as sampling rate updates, where only the latest command is relevant and temporal ordering of previous commands is immaterial.

Notifier Lifecycle

***Table 5.13:* LabVIEW Notifier Operation Summary**

Notifier Primitive	***Function***	***Description***
Obtain Notifier	*Obtain Notifier*	*Creates a named typed notifier. Returns a reference shared between the Main Loop (publisher) and all sensor loops (subscribers).*
Send Notification	*Send Notification*	*Publishes a new sampling rate value. The published value immediately overwrites the previous notification. Called by the Main Control Loop.*
Wait on Notification	*Wait on Notif.*	*Blocks the sensor loop until a new notification is received or a timeout expires. Returns the new sampling rate value.*
Release Notifier	*Release Notifier*	*Destroys the notifier and frees memory. Called once during VI shutdown.*

5.6.3 Independent Sensor Loop Design

In the Enhanced Final Architecture, each sensor loop contains its own Wait on Notification call that blocks until the Main Control Loop publishes a new sampling rate. Upon receiving the updated rate, the sensor loop applies the new wait interval to its timing primitive, immediately adapting its acquisition frequency without requiring a VI restart. The four sensor loops operate simultaneously in parallel, with no synchronization dependency among them — each loop's timing is governed solely by its own received notifier value.

This design yields measurable power savings. At danger Level 0 (SAFE), the soil moisture loop waits 900 ms between acquisitions. At Level 4 (CRITICAL), the same loop samples every 2 ms. The duty cycle ratio between these extremes is 450:1, representing a theoretical power reduction of up to 99.8% for the soil moisture acquisition channel during quiescent operation compared to maximum-rate sampling.

5.6.4 Data Flow Architecture

The complete data flow in the Enhanced Final Architecture proceeds as follows:

- 1. Sensor Acquisition: Each independent sensor loop reads its assigned channel(s) from the corresponding NI WSN node, applies calibration equations, and produces calibrated engineering-unit values.
- 2. Queue Enqueue (Sensor → Main): Each sensor loop enqueues its calibrated reading into a dedicated typed Queue. Queues are unidirectional (sensor → Main Control Loop) and carry only sensor data.
- 3. Queue Dequeue (Main ← Sensor): The Main Control Loop dequeues all pending sensor readings, assembles the TCP packet, and transmits to the Python backend.
- 4. Python Classification: Python receives the packet, classifies the data, updates Firebase, sends alerts if necessary, and returns the danger level integer.
- 5. Notifier Broadcast (Main → Sensor Loops): Based on the received danger level, the Main Control Loop sends updated sampling rate values to all sensor loops via Notifiers, completing the adaptive control feedback loop.

5.7 Comparative Analysis — Preliminary vs. Enhanced Architecture

This section provides a comprehensive engineering comparison between the Preliminary Architecture (common sampling rate) and the Enhanced Final Architecture (independent adaptive sampling rates) to quantify the improvements achieved and justify the design evolution.

Table 5.14: ***Architecture Comparison — Preliminary vs. Enhanced Final Design***

Criterion	***Preliminary Architecture***	***Enhanced Final Architecture***
Number of Loops	*5 (4 sensor + 1 main)*	*5 (4 sensor + 1 main, fully decoupled)*
Sampling Strategy	*Single common rate for all sensors*	*Independent adaptive rate per sensor*
Queue Usage	*Yes — sensor data FIFO*	*Yes — sensor data FIFO (unchanged)*
Notifier Usage	*No*	*Yes — adaptive rate control broadcast*
Sampling Rate Range	*Fixed (e.g., 500 ms)*	*2 ms (CRITICAL) to 1000 ms (SAFE)*
CPU Utilization	*Higher — all loops at max rate*	*Lower during safe conditions*
Power Consumption	*Higher — constant radio Tx*	*Lower — reduced Tx during safe periods*
Response Time	*Fixed latency*	*Minimal latency during danger events*
Scalability	*Low — adding sensor affects all rates*	*High — loops are independent*
Maintainability	*Moderate — shared rate constant*	*High — per-loop configuration*
Reliability	*Good — simple logic*	*Better — fault in one loop does not halt others*
Modularity	*Moderate*	*High — clean Queue/Notifier abstraction*
Expandability	*Limited*	*High — new sensors added as new loops*
Battery Life Impact	*Shorter — maximum duty cycle*	*Longer — adaptive duty cycle*

5.7.1 Engineering Discussion

The transition from the Preliminary to the Enhanced Final Architecture represents a fundamental shift in design philosophy: from a static, synchronous sampling model to a dynamic, event-driven adaptive model. The most significant engineering benefit is the reduction in unnecessary radio transmission cycles during quiescent monitoring periods. In the preliminary design, all four WSN nodes transmit data at the same fixed rate, consuming battery power regardless of the current fire risk level. In the Enhanced Final Architecture, nodes

transmit only as frequently as required by the current danger classification, with the Main Control Loop acting as a real-time resource manager.

The introduction of Notifiers for control signal propagation eliminates the latency that would be introduced if control messages were transmitted via Queues. A Queue-based control channel would require the sensor loop to consume and discard queued data elements until the latest control message was reached, introducing unnecessary processing overhead. Notifiers, by contrast, deliver only the most recent value with zero queuing latency, making them the architecturally correct choice for this application.

The Enhanced Final Architecture also demonstrates superior fault isolation. In the preliminary design, a blocking or error-prone sensor loop could potentially affect the shared sampling rate constant and thus alter the behavior of all other loops. In the enhanced design, each loop's sampling rate is governed by its own received Notifier value, and a fault in one loop does not propagate to others. This architectural isolation improves overall system reliability and simplifies fault diagnosis.

From a scalability perspective, the Enhanced Final Architecture supports the addition of new sensor modalities — for example, optical smoke detectors, infrared flame sensors, or additional thermocouple channels — as self-contained sensor loops that are integrated through the existing Queue and Notifier interfaces, requiring no modifications to the Main Control Loop or the Python backend.

5.8 Experimental Results and Sensor Measurements Sensor Measurements

The following results were obtained from the LabVIEW monitoring interface during real-time operation of the wireless-based forest fire detection system.

State 1: Safe State (Danger Level = 0)

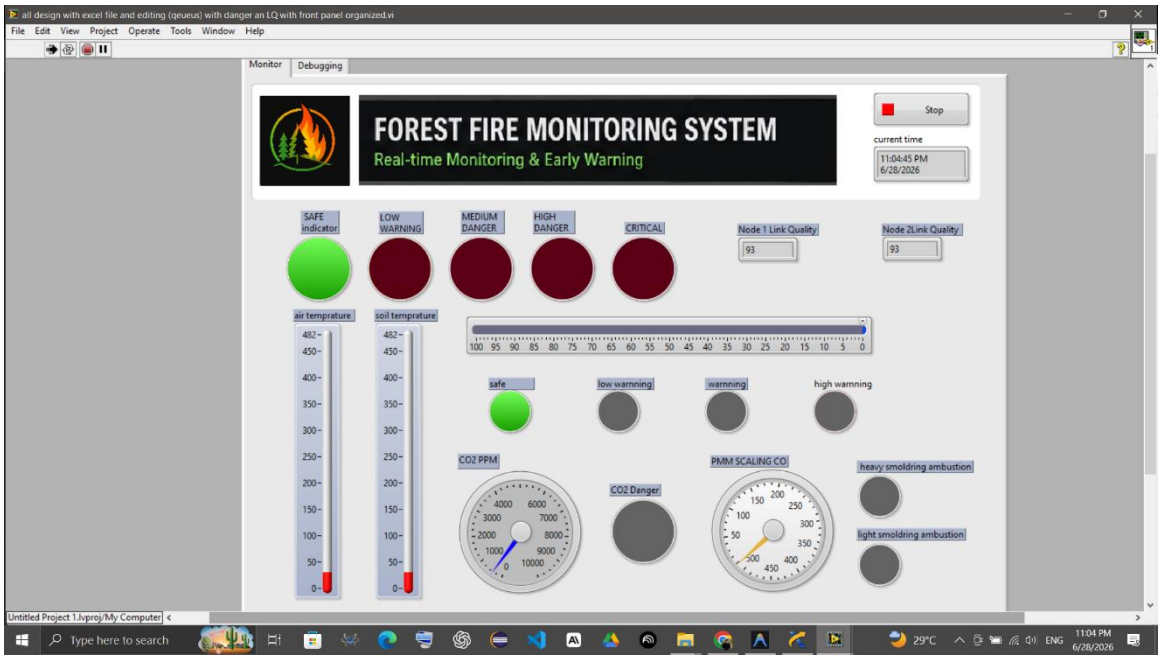


Figure 5.20: Safe state – Labview

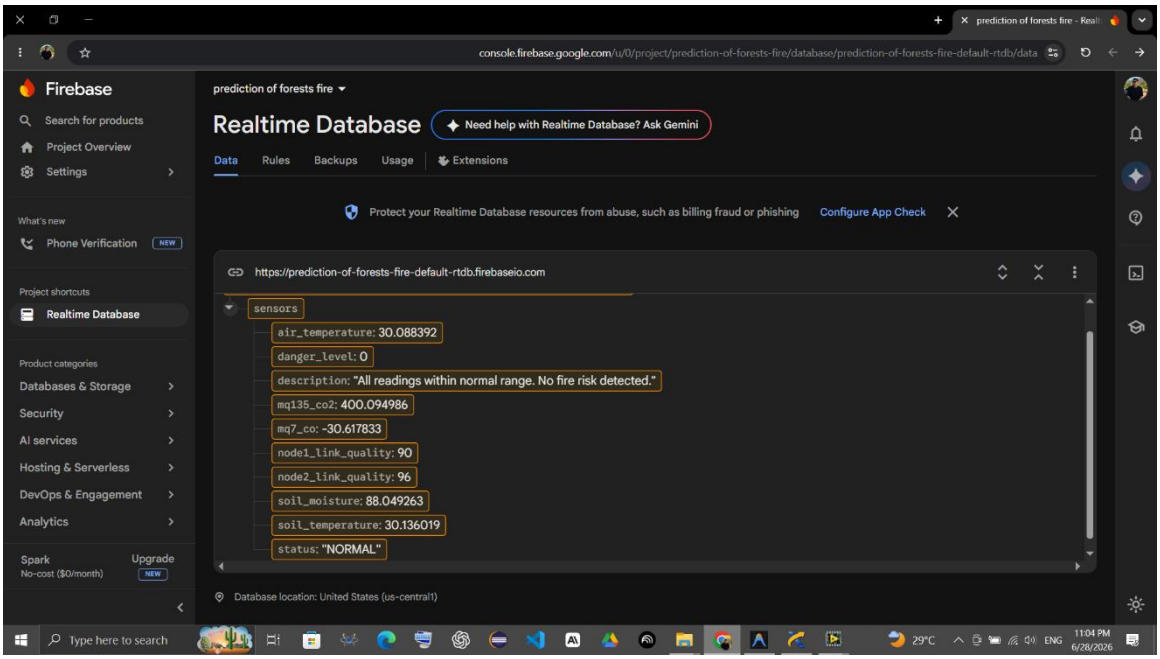


Figure 5.21: Safe state – Firebase

State 2: Low Warning State (Danger Level = 1)

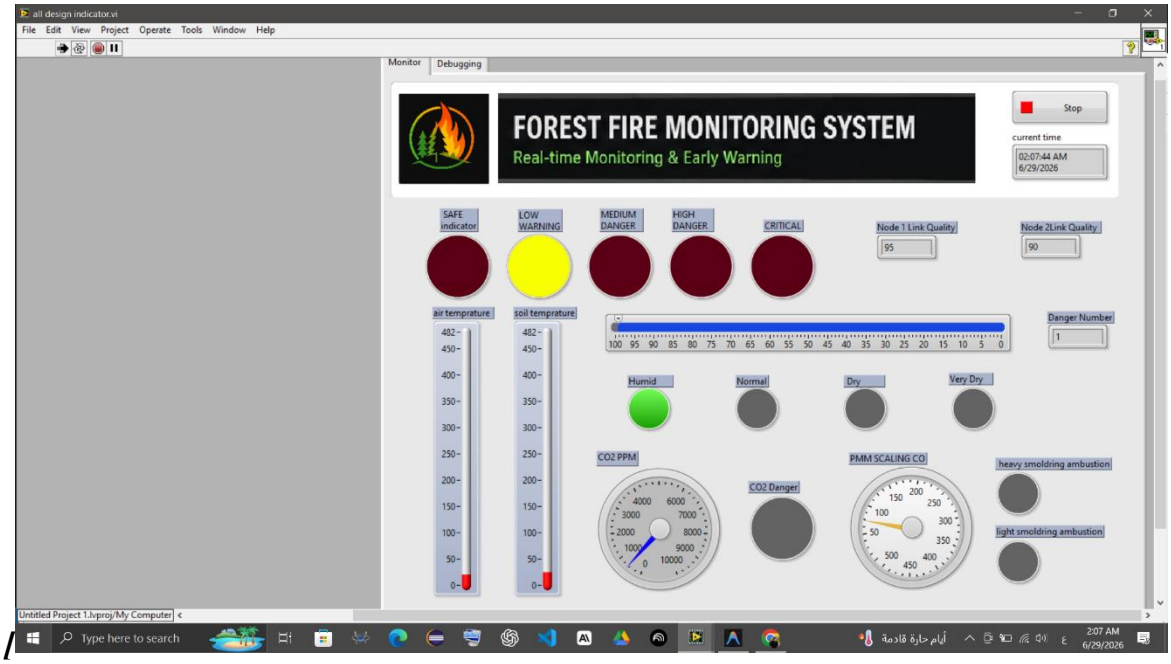


Figure 5.22: Low Warning state – Labview

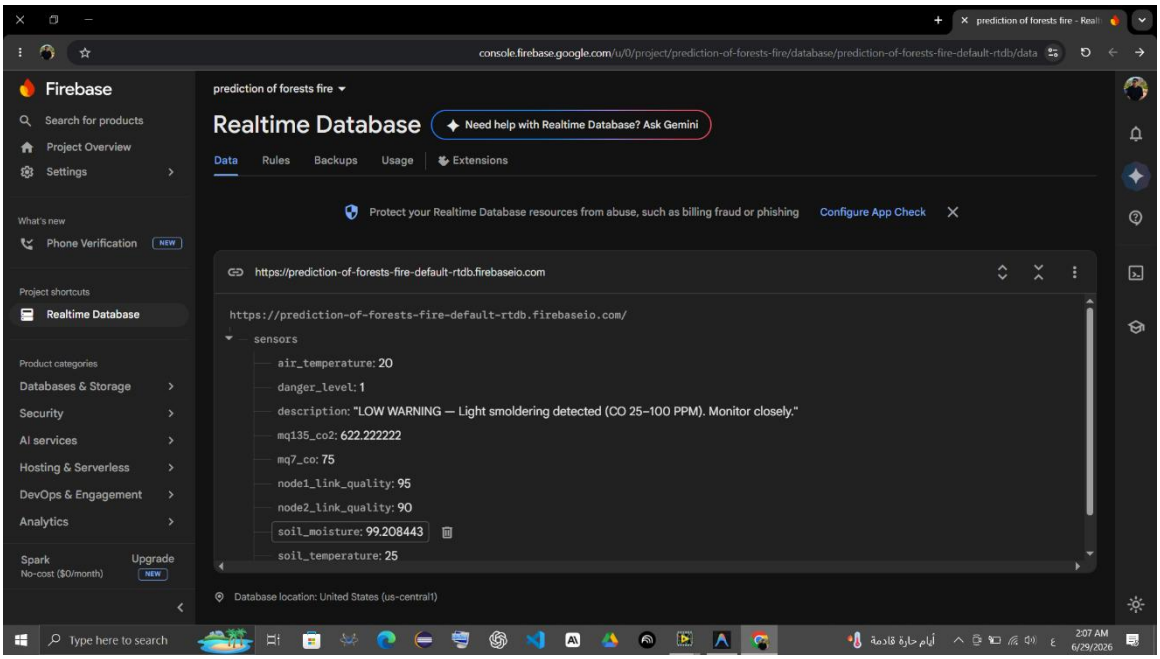


Figure 5.23: Low Warning state – Firebase

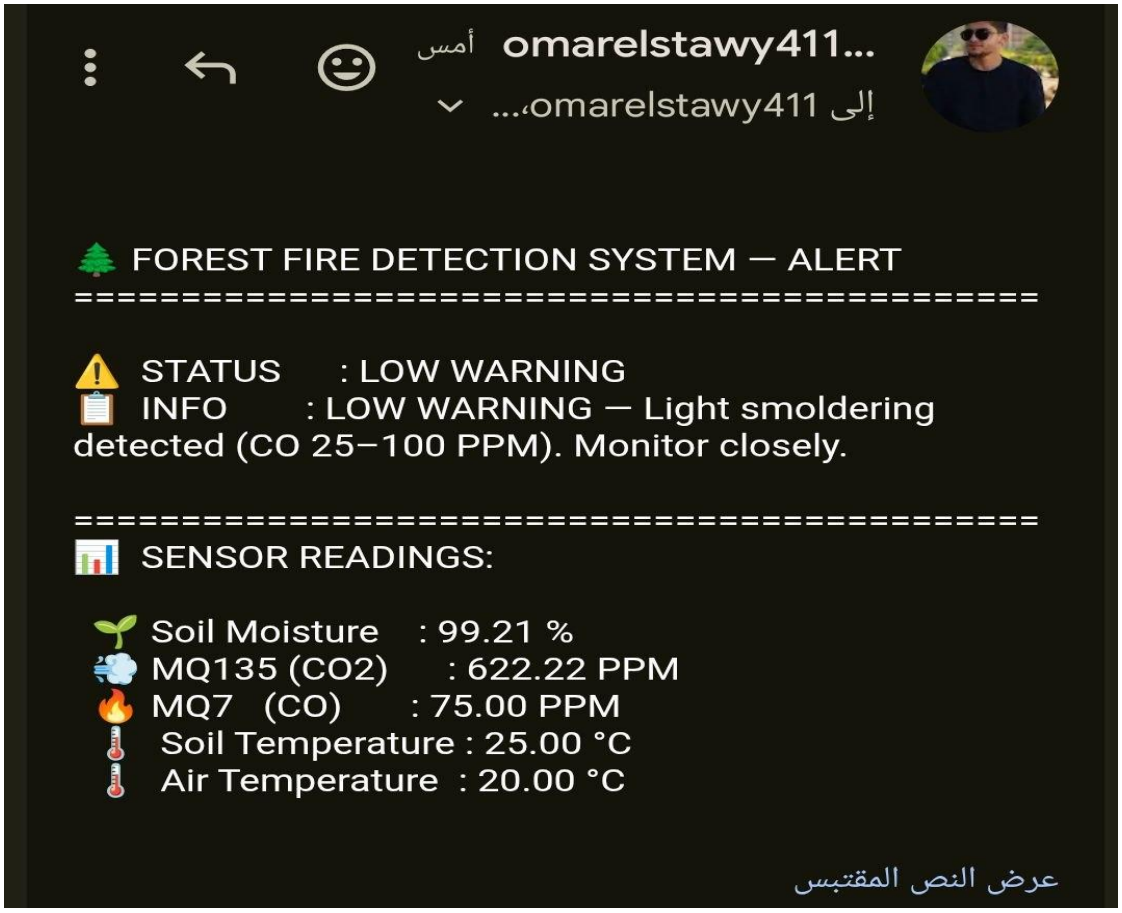


Figure 5.24: Low Warning state – Mail

State 3: Medium Warning State (Danger Level = 2)

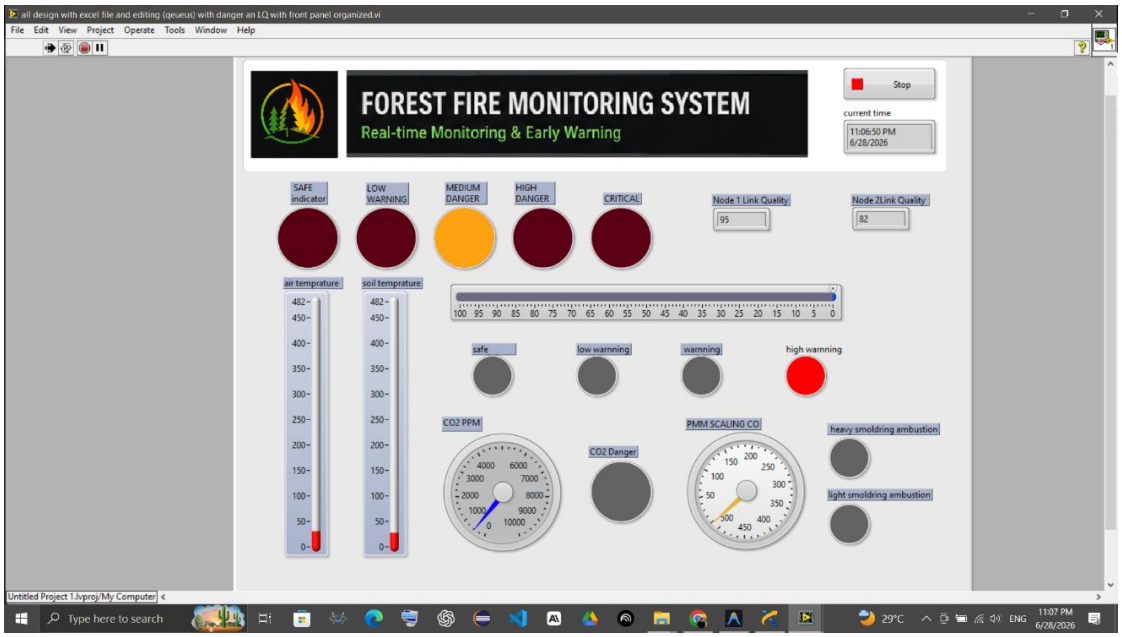


Figure 5.25: Medium Warning state – Labview

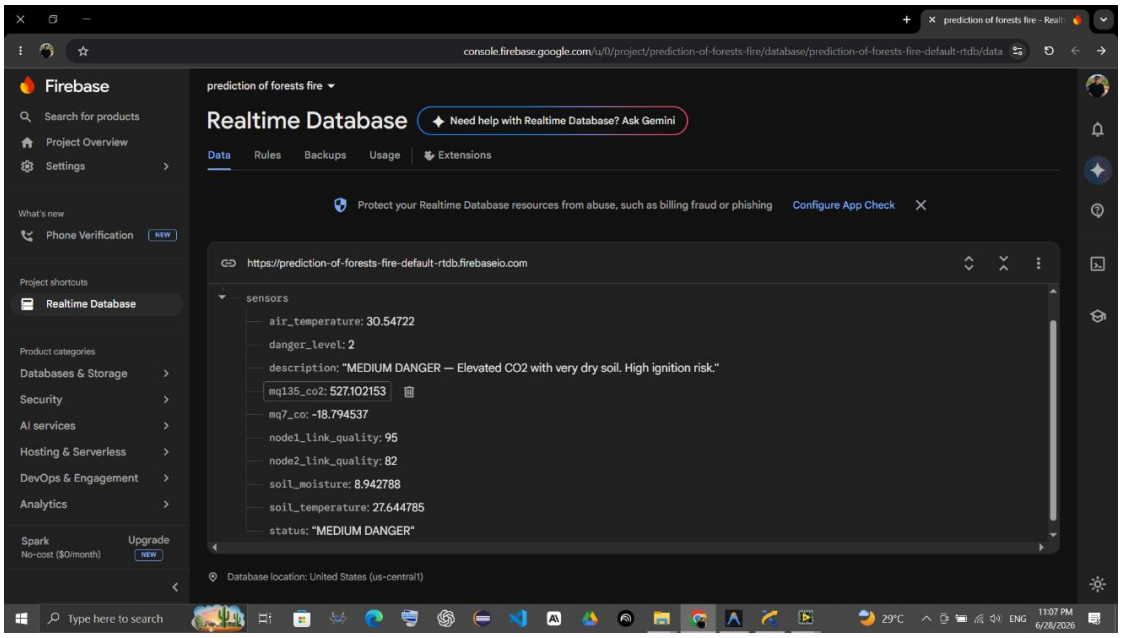


Figure 5.26: Medium Warning state – Firebase

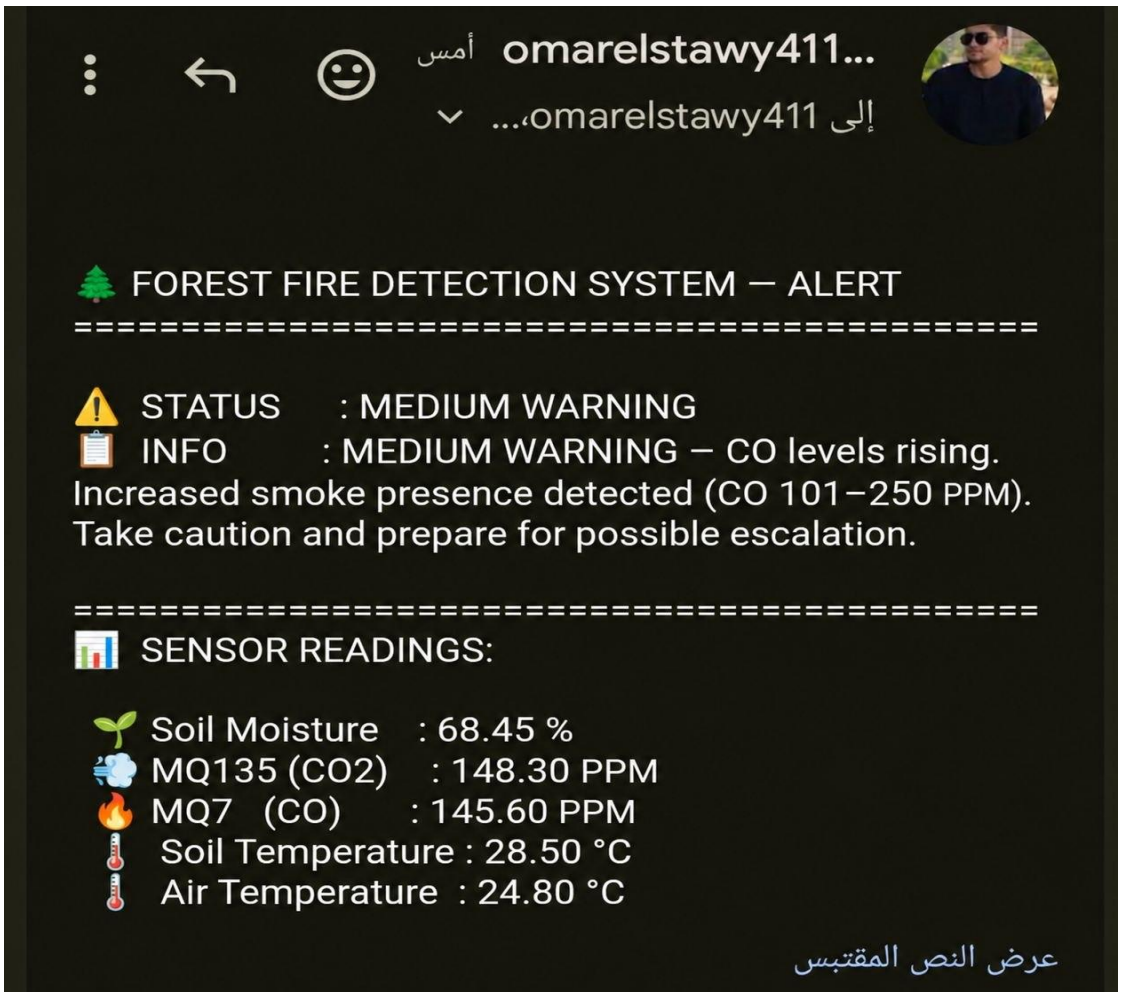

omarelstawy411...
أمس
إلى omarelstawy411...
FOREST FIRE DETECTION SYSTEM — ALERT
STATUS : MEDIUM WARNING
INFO : MEDIUM WARNING — CO levels rising.
Increased smoke presence detected (CO 101–250 PPM).
Take caution and prepare for possible escalation.
SENSOR READINGS:
Soil Moisture : 68.45 %
MQ135 (CO2) : 148.30 PPM
MQ7 (CO) : 145.60 PPM
Soil Temperature : 28.50 °C
Air Temperature : 24.80 °C
عرض النص المقتبس

Figure 5.27: Medium Warning state – Mail

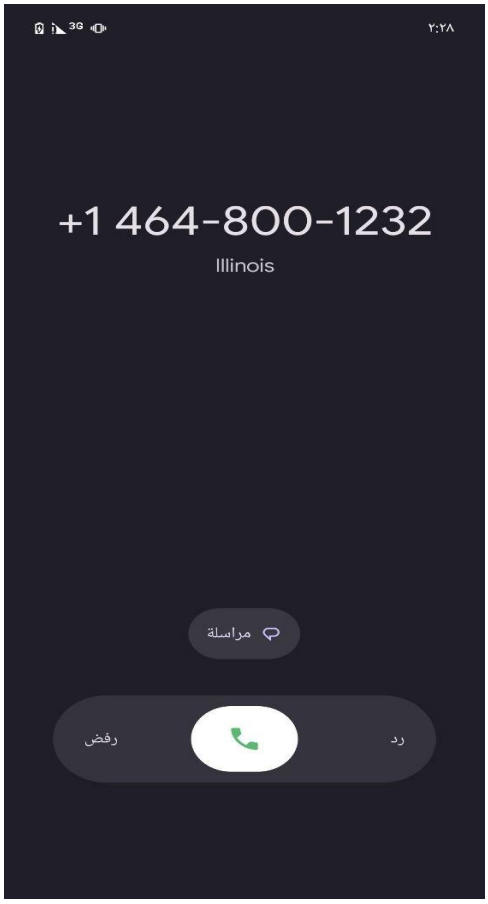


Figure 5.28: Medium Warning state – call

State 3: High Warning State (Danger Level = 3)

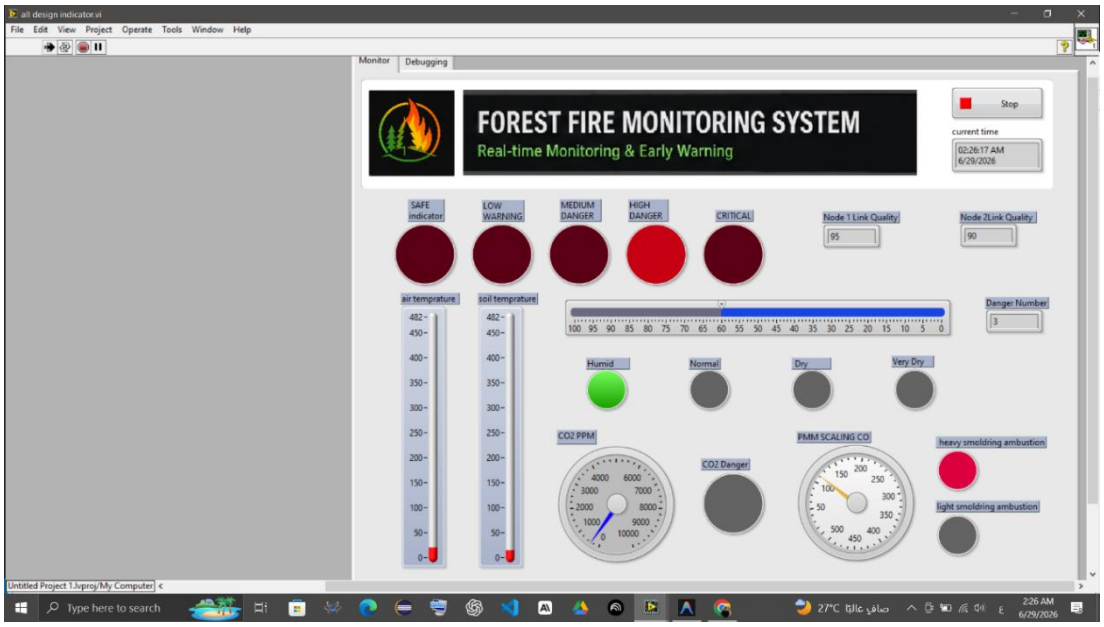


Figure 5.29: High Warning state – Labview

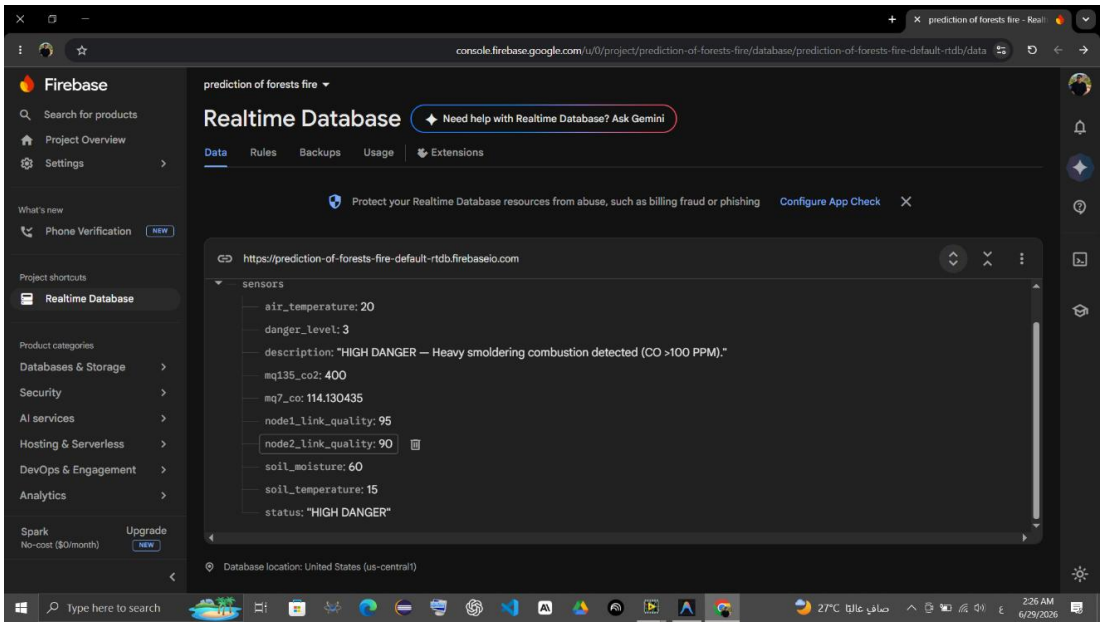


Figure 5.30: High Warning state – Firebase

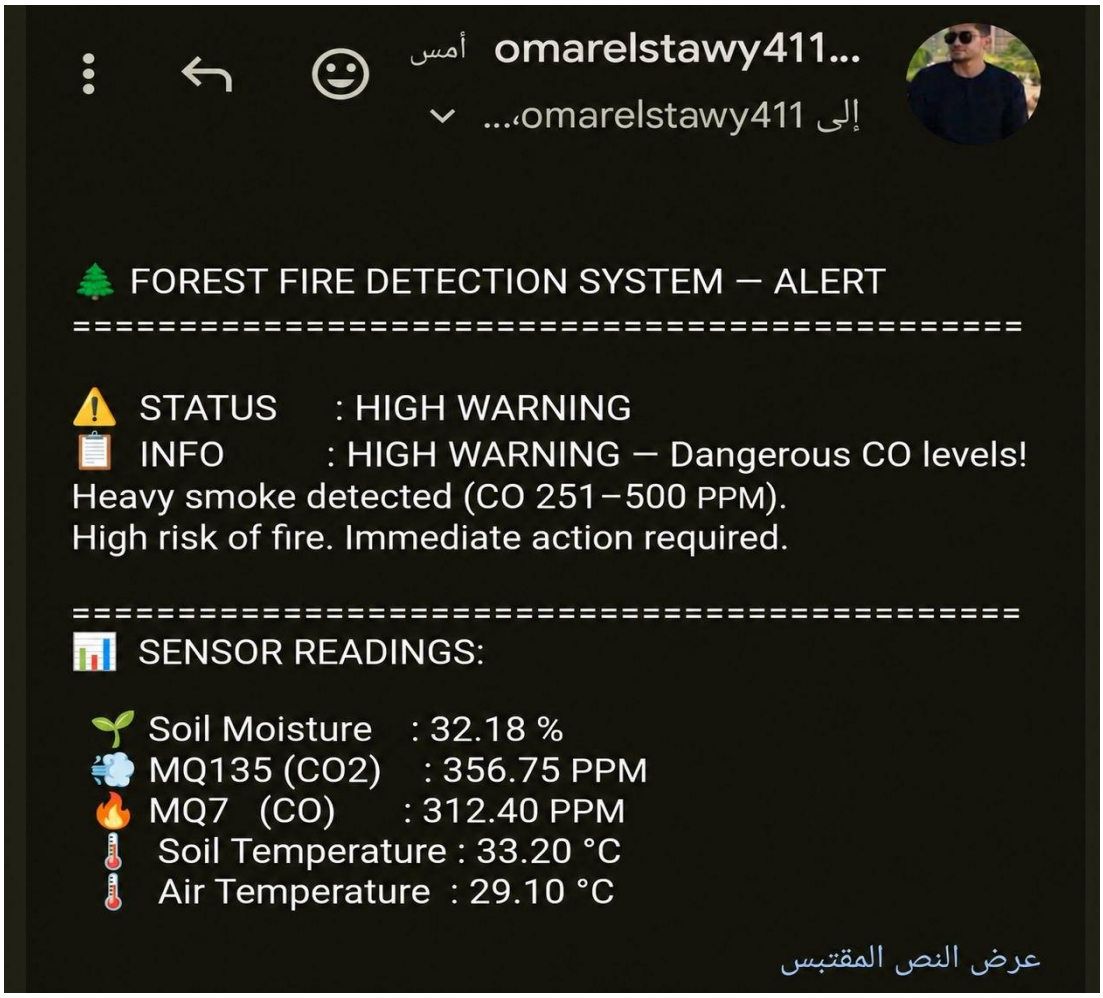

omarelstawy411...
أمس
إلى omarelstawy411...
FOREST FIRE DETECTION SYSTEM — ALERT
STATUS : HIGH WARNING
INFO : HIGH WARNING — Dangerous CO levels!
Heavy smoke detected (CO 251–500 PPM).
High risk of fire. Immediate action required.
SENSOR READINGS:
Soil Moisture : 32.18 %
MQ135 (CO2) : 356.75 PPM
MQ7 (CO) : 312.40 PPM
Soil Temperature : 33.20 °C
Air Temperature : 29.10 °C
عرض النص المقتبس

Figure 5.31: High Warning state – Mail

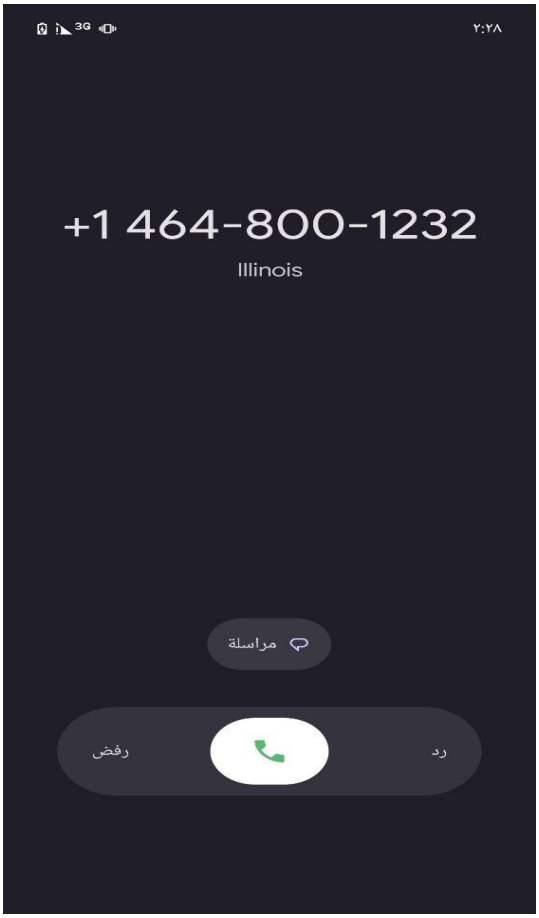


Figure 5.32: High Warning state – call

State 5: Critical State (Danger Level = 4)

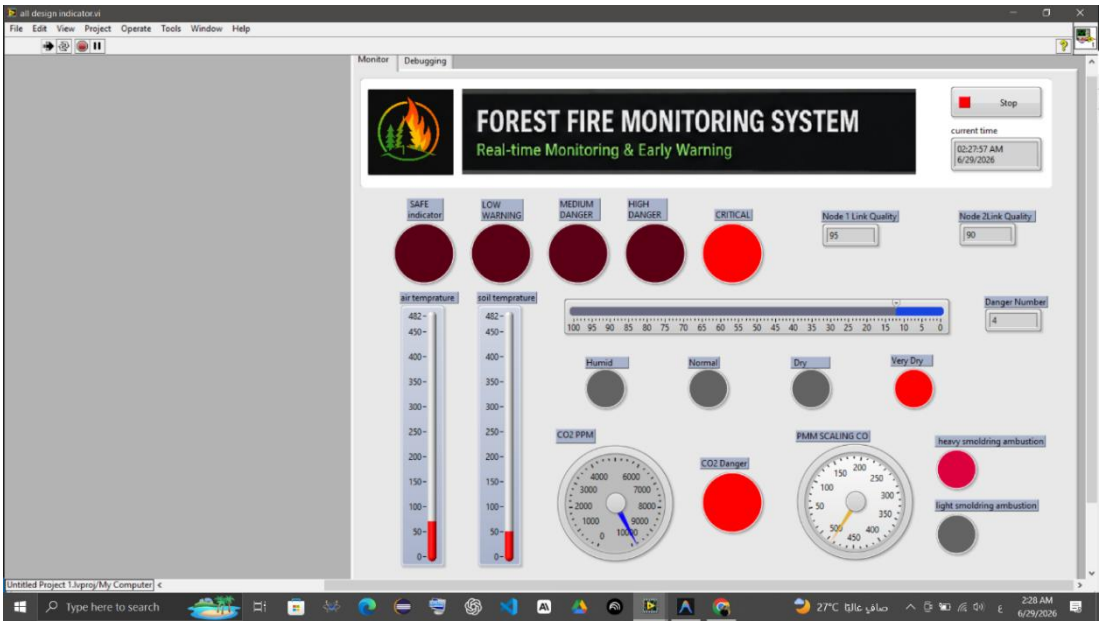


Figure 5.33: Critical state – Labview

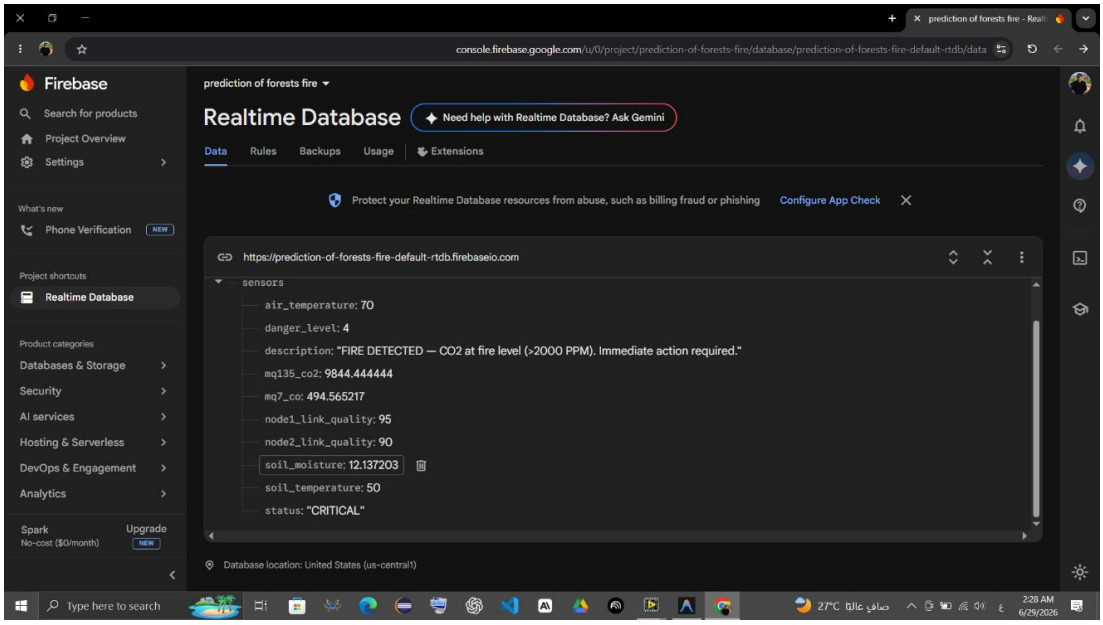


Figure 5.34: ***Critical state – Firebase***

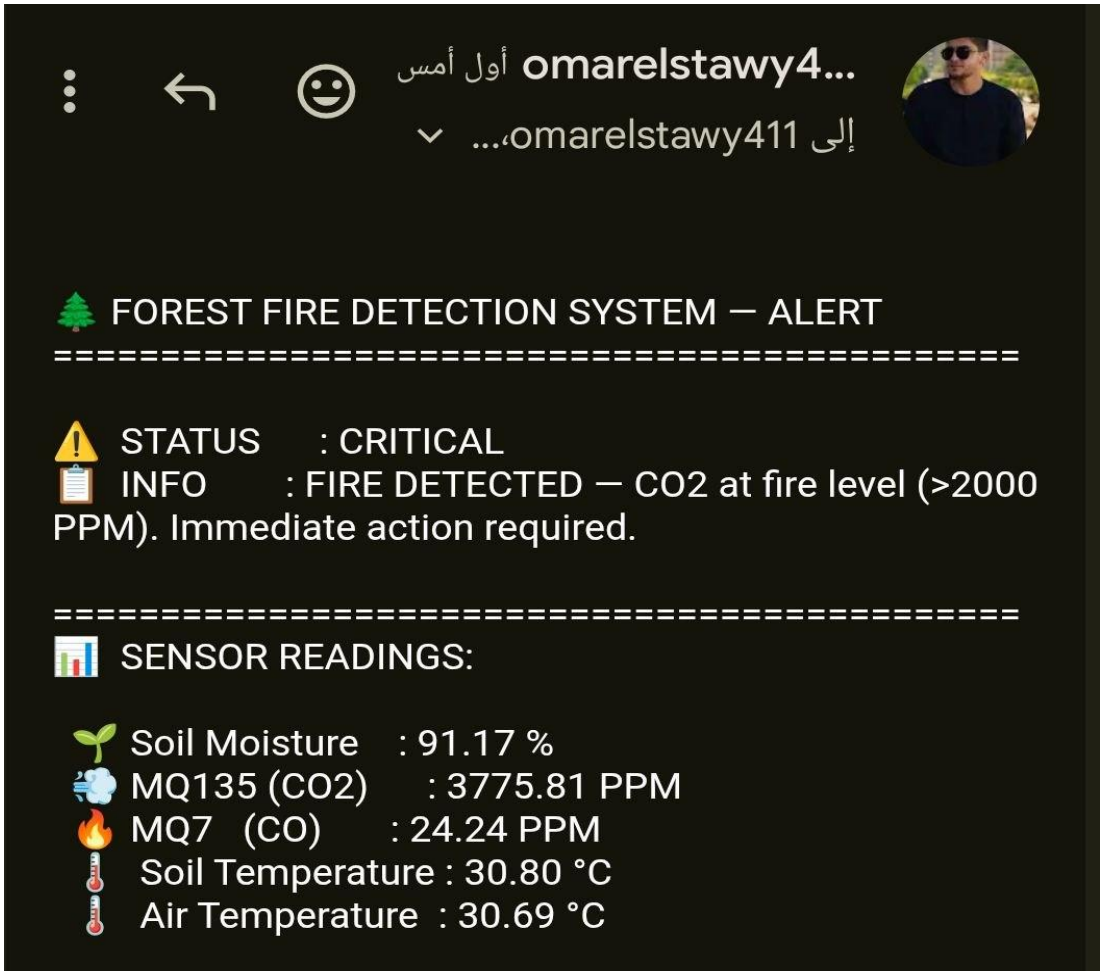

omarelstawy4...
أول أمس
إلى omarelstawy411...
FOREST FIRE DETECTION SYSTEM — ALERT
STATUS : CRITICAL
INFO : FIRE DETECTED — CO2 at fire level (>2000 PPM). Immediate action required.
SENSOR READINGS:
Soil Moisture : 91.17 %
MQ135 (CO2) : 3775.81 PPM
MQ7 (CO) : 24.24 PPM
Soil Temperature : 30.80 °C
Air Temperature : 30.69 °C

Figure 5.35: ***Critical state – Mail***

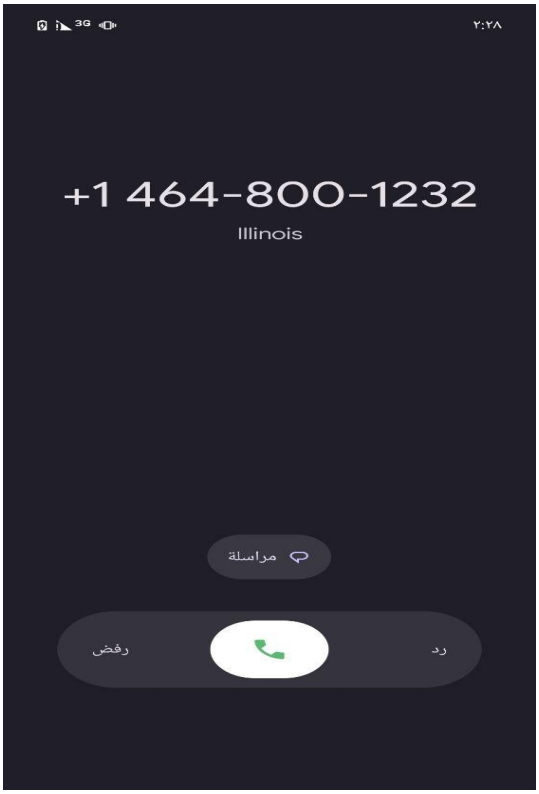


Figure 5.36: Critical state – call

State 5: Connection loss state

omarelstawy41...
قبل ٤ أيام
إلى أنا، ahmed0106603...
SENSOR CONNECTIVITY ALERT
The following sensor(s) returned invalid readings (nan/inf):
air_temperature
Possible causes:
- Sensor disconnected or damaged
- Wiring / power issue
- LabVIEW signal conversion error
Please check your sensor connections.
This is an automated alert from your WSN system.

Figure 5.37: Connection loss state – Mail

Excel Data Logging

The proposed system automatically exports sensor measurements along with their timestamps to an Excel spreadsheet, creating a well-organized historical database of environmental conditions. This exported dataset facilitates data visualization through charts and dashboards, enabling users to monitor sensor behavior and observe environmental trends over time. Furthermore, the recorded data can be utilized for advanced data analytics, statistical processing, anomaly detection, and predictive modeling using Artificial Intelligence (AI) and Machine Learning (ML) techniques. The Excel format also ensures compatibility with various data analysis platforms, allowing researchers to perform further analysis and generate meaningful insights for future system improvements.

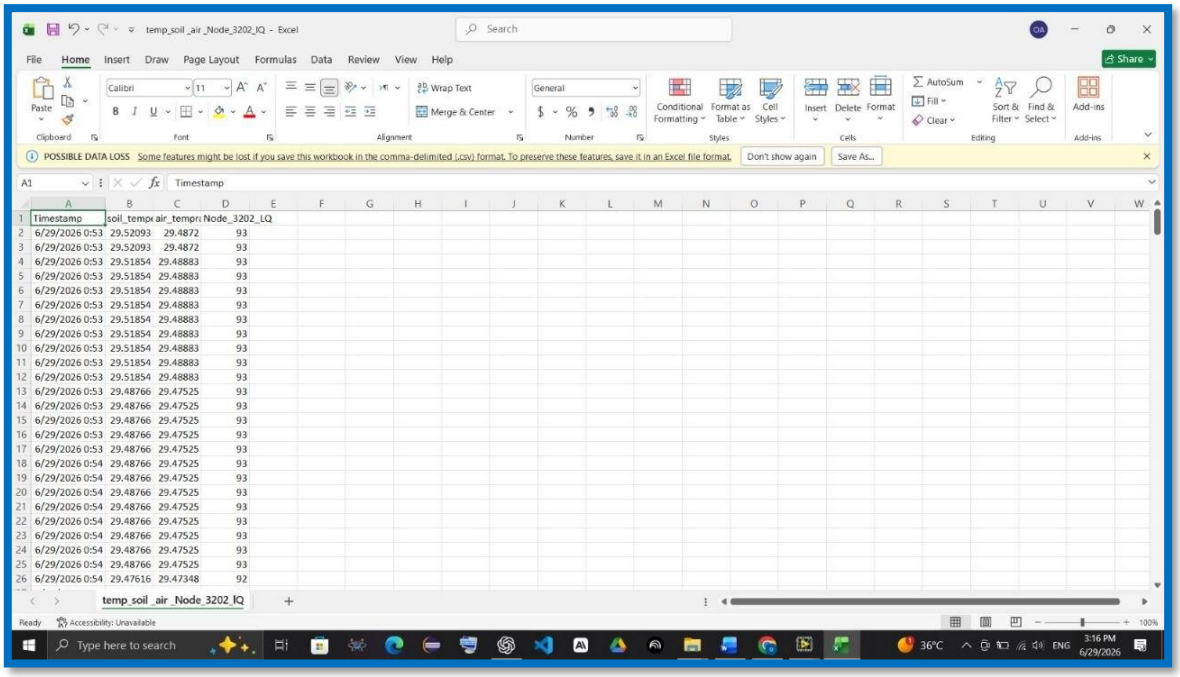


*Figure 5.38: **Excel-Based Sensor Data Logging***

5.9 Power Consumption and Battery Life Analysis

5.9.1 System Overview

The WSN consists of two NI WSN nodes powered by four Energizer AA non-rechargeable batteries each (1.5 V, 2,850 mAh rated; total pack: 6 V, 2,850 mAh usable). Node NI 3202 monitors CO, CO_2, and Soil Moisture; Node NI 3212 monitors Air and Soil Temperature. Link-quality index (LQI) is transmitted with every CO sample (3202) and every temperature sample (3212), adding negligible overhead (~2 bytes per frame).

Datasheet values used where available (NI WSN-3202 / 3212 specifications). Assumed values are conservative engineering estimates consistent with IEEE 802.15.4-class nodes.

5.9.2 Node Current Consumption and Battery Life

***Table 5.15:** Node current consumption summary. (* = estimated)*

Parameter	***Sleep (µA)***	***Sensor Acq (mA)***	***TX (mA)***	***RX (mA)***	***Duty Cycle Safe (%)***	***Duty Cycle Crit (%)***	***I_avg Safe (µA)***	***I_avg Crit (µA)***
NI WSN-3202	*35**	*3.2**	*12.5**	*10.8**	*0.42*	*10.2*	*86*	*1,320*
NI WSN-3212	*35**	*2.5**	*12.5**	*10.8**	*0.28*	*8.7*	*68*	*1,080*

5.9.3 Key Equations

Average current draw per node (Equation 1):

$$I\_avg = D\_tx \cdot I\_tx + D\_rx \cdot I\_rx + D\_sense \cdot I\_sense + (1 - D\_tx - D\_rx - D\_sense) \cdot I\_sleep$$

Battery life in hours (Equation 2):

$$T\_battery = C\_battery / I\_avg = 2{,}850\ mAh / I\_avg$$

Power-saving percentage vs. critical state (Equation 3):

$$Savings\ (\%) = [(I\_avg\_crit - I\_avg\_safe) / I\_avg\_crit] \times 100$$

5.9.4 Fixed vs. Adaptive Sampling — Comparative Analysis

Table 5.16: Fixed vs. Adaptive Sampling strategy comparison. Fixed rate assumes 1-minute intervals for all sensors.

Parameter	*Fixed Sampling*	*Adaptive (Safe, L0)*	*Adaptive (Med, L2)*	*Adaptive (Crit, L4)*	*Improvement (L0 vs Fixed)*
CO / Temp Interval	*1 min*	*15 min*	*1 min*	*2 sec*	*15× longer*
Soil Moisture Interval	*1 min*	*6 hr*	*1 hr*	*5 min*	*360× longer*
I avg – NI 3202	*~650 µA*	*~86 µA*	*~310 µA*	*~1,320 µA*	*86% lower*
I_avg – NI 3212	*~520 µA*	*~68 µA*	*~250 µA*	*~1,080 µA*	*87% lower*
Battery Life – 3202	*~183 d*	*~1,382 d*	*~384 d*	*~90 d*	*7.5× longer*
Battery Life – 3212	*~229 d*	*~1,747 d*	*~475 d*	*~110 d*	*7.6× longer*

5.9.5 Battery Life Results Summary

Best-case (Safe, Level 0): NI 3202 ≈ 1,382 days (~3.8 years); NI 3212 ≈ 1,747 days (~4.8 years).

Worst-case (Critical, Level 4): NI 3202 ≈ 90 days; NI 3212 ≈ 110 days.

Power saving (Safe vs. Critical): NI 3202 ≈ 93.5% | NI 3212 ≈ 93.7%.

Battery-life improvement (Adaptive Safe vs. Fixed 1-min): ~7.5× for NI 3202; ~7.6× for NI 3212.

5.9.6 Adaptive Sampling Decision Flowchart

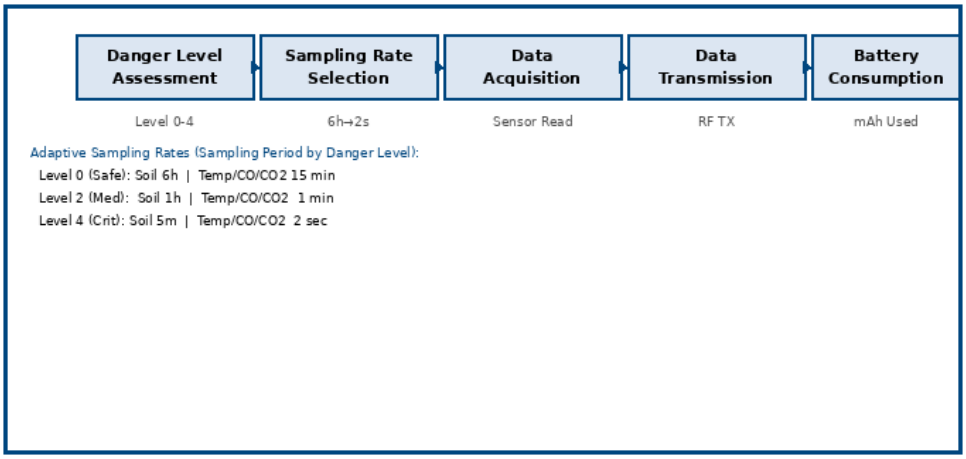

Danger Level Assessment
Sampling Rate Selection
Data Acquisition
Data Transmission
Battery Consumption
Level 0-4
6h→2s
Sensor Read
RF TX
mAh Used
Adaptive Sampling Rates (Sampling Period by Danger Level):
Level 0 (Safe): Soil 6h | Temp/CO/CO2 15 min
Level 2 (Med): Soil 1h | Temp/CO/CO2 1 min
Level 4 (Crit): Soil 5m | Temp/CO/CO2 2 sec

Figure 5.39: Adaptive Sampling Decision and Energy Flow.

5.9.7 Discussion

Benefits of Adaptive Sampling: The five-level adaptive strategy dynamically adjusts sampling intervals from 2-second (Critical) to 6-hour (Safe) cycles. During normal environmental conditions, node activity drops to under 7% of its critical-state duty cycle, slashing average current by more than 93%. This single design decision extends battery

lifetime from ~3 months (worst case) to nearly 4–5 years (best case), eliminating or drastically reducing field maintenance visits.

Power-Saving Mechanism: Energy savings stem from three compounded effects: (i) fewer sensor excitation pulses, (ii) fewer RF transmit bursts — the dominant current draw at 12.5 mA — and (iii) longer idle periods in the sub-40 µA sleep state. Because LQI piggybacks on existing CO/temperature frames, it adds zero extra transmissions.

Trade-off — Responsiveness vs. Energy: The sole cost of extended sleep intervals is detection latency. In Level 0 (Safe), a hazardous CO spike may go undetected for up to 15 minutes. The system mitigates this by escalating through intermediate warning levels (Low → Medium → High) before reaching Critical, each halving or better the latency. At Level 4, 2-second polling ensures near-real-time hazard visibility. For the target underground mining / agricultural use case, this trade-off is engineering-optimal: energy is conserved when conditions are benign and responsiveness is maximised precisely when it matters most.

5.10 Wireless Communication Range Test

To evaluate the communication performance of the wireless sensor network, a practical range test was conducted under different environmental conditions using sensor nodes powered by partially used batteries.

In the first scenario, the communication path contained multiple obstacles, including walls, rooms, and various indoor obstructions. Under these conditions, the maximum reliable communication distance achieved between the wireless node and the gateway was approximately 45.7 meters.

In the second scenario, the communication path contained significantly fewer obstacles and was mainly composed of open space with limited vegetation and trees. Under these conditions, the maximum reliable communication distance increased to approximately 87 meters.

The results demonstrate the significant impact of environmental obstacles on wireless signal propagation. Walls and indoor structures introduce signal attenuation and reduce the effective communication range, whereas open environments allow the wireless signal to travel greater distances with lower losses.

It should also be noted that these measurements were obtained while using batteries that were not fully charged and had already been partially consumed. Therefore, it is expected that the communication range may be further improved when operating with new batteries or a stable external power supply.

According to the manufacturer specifications, the theoretical communication range of the wireless sensor nodes can reach approximately 300 meters or more under ideal line-of-sight conditions, where minimal interference and obstacles exist. However, practical deployment environments rarely provide such ideal conditions. Therefore, the measured ranges of 45.7 meters and 87 meters are considered realistic results that reflect actual operating conditions in the presence of physical obstacles, vegetation, environmental interference, and partially depleted batteries.

These findings provide valuable insight into the practical deployment limitations of the proposed wireless sensor network and can be used to optimize node placement in future implementations. The results also confirm that the system can maintain reliable communication under realistic field conditions, making it suitable for environmental monitoring and early forest fire detection applications.

5.11 Chapter Summary

This chapter presented the complete implementation of the proposed Early Forest Fire Detection and Prediction System, covering both the LabVIEW-based data acquisition layer and the Python-based intelligent processing backend. The implementation demonstrated the integration of wireless sensor nodes, real-time TCP communication, sensor calibration, data logging, and multi-level fire danger classification. The developed system also incorporated cloud-based monitoring through Firebase, automated email and voice-call notifications, and Excel-based data logging for future visualization and AI-driven analysis. Furthermore, the chapter introduced an enhanced adaptive sampling architecture using Queue and Notifier mechanisms, significantly improving system responsiveness, scalability, and energy efficiency compared to the preliminary design. Finally, experimental results validated the system under all danger levels, while the power consumption analysis confirmed that the adaptive sampling strategy substantially extends battery lifetime without compromising real-time fire detection performance.

CHAPTER

6

Conclusion & Future Work

6.1 Conclusion

This project presented the design and implementation of a comprehensive Early Forest Fire Detection and Prediction System based on a Wireless Sensor Network (WSN) architecture. The proposed system integrates NI WSN sensor nodes, LabVIEW, Python, TCP/IP communication, Firebase cloud services, adaptive sampling, and automated notification mechanisms to provide reliable real-time environmental monitoring and early fire detection.

The wireless sensor nodes continuously monitor critical environmental parameters, including soil moisture, carbon monoxide (CO), carbon dioxide (CO_2), soil temperature, and air temperature. The collected data are transmitted through the NI WSN gateway to the LabVIEW application, where they are calibrated, processed, visualized, and forwarded to the Python backend for intelligent danger classification.

To improve the efficiency of the monitoring process, an adaptive sampling strategy was implemented. Instead of operating all sensors at a fixed sampling rate, the system dynamically adjusts the sampling interval according to the detected danger level. This approach significantly reduces energy consumption during normal environmental conditions while providing rapid data acquisition whenever a potential fire hazard is detected, thereby extending the operational lifetime of the wireless sensor nodes.

The Python backend performs multi-parameter danger classification using predefined environmental thresholds and automatically updates the Firebase Realtime Database, enabling cloud-based monitoring. Furthermore, the system generates automated email notifications and voice-call alerts whenever abnormal environmental conditions are detected, ensuring that emergency personnel receive immediate warnings without requiring continuous manual supervision.

To support future research and advanced analytics, all sensor measurements are automatically exported to Excel files with timestamps, creating a structured historical database suitable for data visualization, statistical analysis, dashboard development, and Artificial Intelligence (AI) or Machine Learning (ML) applications.

Experimental evaluation demonstrated that the proposed system successfully operates under all predefined danger levels, providing reliable communication, accurate monitoring, efficient adaptive sampling, and timely alert generation. The modular software architecture also allows future expansion through the addition of new sensors, intelligent algorithms, and communication technologies without major modifications.

Overall, the proposed system provides a scalable, reliable, energy-efficient, and cost-effective solution for early forest fire detection and prediction. By combining wireless sensing, intelligent processing, cloud connectivity, adaptive resource management, and automated alerting, the developed platform represents a strong foundation for next-generation smart environmental monitoring systems.

6.2 Future Work

Although the proposed system demonstrates effective real-time forest fire detection and monitoring, several enhancements can be considered to further improve its performance, scalability, and intelligence.

Artificial Intelligence and Machine Learning: Train AI and ML models using the historical sensor data stored in Excel files and cloud databases to predict forest fire incidents before critical conditions are reached. Intelligent models can continuously learn from environmental patterns, improving prediction accuracy while reducing false alarms.

Advanced Data Analytics: Develop predictive analytics dashboards capable of identifying long-term environmental trends, seasonal changes, and abnormal sensor behavior to support proactive decision-making.

Satellite and Remote Sensing Integration: Integrate satellite imagery and thermal remote sensing data with ground sensor measurements to provide comprehensive monitoring over large forest areas and improve fire localization accuracy.

GPS-Based Localization: Equip each wireless sensor node with a GPS module to transmit its geographical coordinates together with sensor readings, allowing emergency teams to identify the exact location of fire incidents immediately.

Dedicated Mobile Application: Develop Android and iOS applications that provide real-time monitoring, interactive dashboards, push notifications, and remote system management from anywhere.

Drone (UAV) Integration: Deploy autonomous drones equipped with thermal cameras and environmental sensors to verify detected alarms, estimate fire size, inspect affected areas, and provide live aerial images for emergency response teams.

LoRa/LoRaWAN Communication: Replace or complement the current wireless communication technology with LoRa or LoRaWAN to increase communication range while maintaining ultra-low power consumption, making the system suitable for very large forests.

Cloud-Based Analytics: Expand the current cloud platform to support long-term data storage, advanced visualization dashboards, remote configuration, and multi-user monitoring through secure web applications.

Scalability Testing: Evaluate system performance using a significantly larger number of wireless sensor nodes distributed across extensive monitoring areas to assess network scalability, communication latency, and reliability.

Advanced Routing Protocols: Investigate intelligent Ad Hoc routing protocols and self-healing communication mechanisms to improve network reliability, fault tolerance, and load balancing in large-scale deployments.

Energy Harvesting: Integrate solar panels and energy harvesting technologies to enable autonomous operation and significantly reduce battery replacement and maintenance costs.

Cybersecurity Enhancements: Strengthen system security by implementing secure communication protocols, encryption, authentication mechanisms, and intrusion detection techniques to protect both sensor data and cloud services.

Hybrid Intelligent Decision Support: Replace the current rule-based danger classification with hybrid AI techniques such as fuzzy logic, deep learning, or reinforcement learning to improve adaptability under different environmental and climate conditions.

Support for Mobile Sensor Nodes: Extend the system to support mobile wireless sensor nodes, allowing dynamic topology changes and deployment in emergency response, disaster recovery, and other rapidly changing environments.

Expansion of Environmental Sensors: Integrate additional sensors such as smoke, flame, wind speed, wind direction, atmospheric pressure, rainfall, and solar radiation sensors to increase detection accuracy and provide a more comprehensive assessment of environmental conditions.

By integrating wireless sensor networks with cloud computing, Artificial Intelligence, satellite monitoring, GPS localization, renewable energy technologies, and autonomous aerial monitoring, the proposed system can evolve into a fully intelligent smart forest monitoring platform capable of early detection, prediction, precise localization, and efficient management of forest fire incidents.

References

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless Sensor Networks: A Survey," *Computer Networks*, vol. 38, no. 4, pp. 393–422, 2002.

[2] H. Karl and A. Willig, *Protocols and Architectures for Wireless Sensor Networks*. Chichester, UK: John Wiley & Sons, 2005.

[3] J. Zheng and A. Jamalipour, *Wireless Sensor Networks: A Networking Perspective*. Hoboken, NJ, USA: Wiley-IEEE Press, 2009.

[4] G. J. Pottie and W. J. Kaiser, "Wireless Integrated Network Sensors," *Communications of the ACM*, vol. 43, no. 5, pp. 51–58, 2000.

[5] P. Gburzynski and I. Nikolaidis, "Rule-Driven Forwarding for Resilient WSN Infrastructures," *Sensors*, vol. 22, no. 22, Art. no. 8708, 2022.

[6] S.-H. Yang, *Wireless Sensor Networks: Principles, Design and Applications*. London, UK: Springer, 2014.

[7] A. Bachir, M. Dohler, T. Watteyne, and K. K. Leung, "MAC Essentials for Wireless Sensor Networks," *IEEE Communications Surveys & Tutorials*, vol. 12, no. 2, pp. 222–248, 2010.

[8] C.-Y. Wan, A. T. Campbell, and L. Krishnamurthy, "PSFQ: A Reliable Transport Protocol for Wireless Sensor Networks," in *Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications (WSNA)*, Atlanta, GA, USA, 2002, pp. 1–11.

[9] K. Sohraby, D. Minoli, and T. Znati, *Wireless Sensor Networks: Technology, Protocols, and Applications*. Hoboken, NJ, USA: John Wiley & Sons, 2007.

[10] J. Yick, B. Mukherjee, and D. Ghosal, "Wireless Sensor Network Survey," *Computer Networks*, vol. 52, no. 12, pp. 2292–2330, 2008.

[11] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015.

[12] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson, "Wireless Sensor Networks for Habitat Monitoring," in *Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications (WSNA)*, Atlanta, GA, USA, 2002.

[13] National Instruments, *NI WSN-9791 Ethernet Gateway User Guide and Specifications*. National Instruments Corporation.

[14] National Instruments, *NI WSN-3202 Voltage Input Node User Guide and Specifications*. National Instruments Corporation.

[15] National Instruments, *NI WSN-3212 Thermocouple Input Node User Guide and Specifications*. National Instruments Corporation.

[16] National Instruments, *LabVIEW User Manual*. National Instruments Corporation.

[17] National Instruments, *LabVIEW Help: TCP/IP Communication Functions, Queues, Notifiers, and Producer–Consumer Design Pattern*. National Instruments Documentation.

[18] IEEE Standards Association, *IEEE Standard 802.15.4™: Low-Rate Wireless Personal Area Networks (LR-WPANs).*

[19] Python Software Foundation, *Python 3 Documentation.*

[20] Google, *Firebase Realtime Database Documentation.*

[21] Twilio Inc., *Twilio Programmable Voice API Documentation.*

[22] Microsoft Corporation, *Microsoft Excel Documentation.*

[23] Microsoft Corporation, *Microsoft Power BI Documentation.*

[24] MQ Sensors, *MQ-7 Carbon Monoxide Gas Sensor Datasheet.*

[25] MQ Sensors, *MQ-135 Air Quality / Carbon Dioxide Gas Sensor Datasheet.*

[26] International Organization for Standardization (ISO), *ISO 8601: Date and Time—Representations for Information Interchange.*

[27] NASA Earth Observatory, *Wildfire Monitoring and Remote Sensing Resources.*

[28] U.S. Geological Survey (USGS), *Remote Sensing Applications for Wildfire Detection.*

[29] Food and Agriculture Organization of the United Nations (FAO), *Global Forest Resources Assessment.*

[30] IEEE Xplore Digital Library, *Research Articles on Wireless Sensor Networks, Environmental Monitoring, and Forest Fire Detection.*

[31] Elsevier ScienceDirect, *Research Articles on Forest Fire Detection, Wireless Sensor Networks, and Internet of Things Applications.*

[32] SpringerLink Digital Library, *Research Articles on Wireless Sensor Networks and Environmental Monitoring.*

[33] ACM Digital Library, *Research Articles on Wireless Sensor Networks, IoT, and Ad Hoc Networks.*

[34] Google Scholar, *Academic Search Engine for Scientific Literature.*

تمهيد

تُعد شبكات الاستشعار اللاسلكية من أبرز التقنيات الحديثة التي أحدثت نقلة نوعية في مجال المراقبة وجمع البيانات، لما تتميز به من القدرة على استشعار المتغيرات البيئية ونقلها ومعالجتها بصورة آنية، مع استهلاك منخفض للطاقة وإمكانية الانتشار في البيئات الواسعة والصعبة. وقد أسهمت هذه المزايا في توسيع نطاق استخدام هذه الشبكات لتشمل العديد من المجالات، مثل المراقبة البيئية، والزراعة الذكية، والصناعة، والرعاية الصحية، والمدن الذكية، وكان من بين أهم تطبيقاتها أنظمة الكشف المبكر عن حرائق الغابات

.

وانطلاقًا من أهمية هذا التطبيق، يتناول هذا الكتاب تصميم وتنفيذ نظام للكشف عن حرائق الغابات بالاعتماد على شبكات الاستشعار اللاسلكية، حيث يعتمد النظام على مراقبة مجموعة من المتغيرات البيئية، وتحليلها لتحديد مستوى خطورة الحريق، مع توفير آلية للمراقبة اللحظية وإرسال التنبيهات عند اكتشاف أي مؤشرات تدل على احتمالية نشوب الحريق، وذلك من خلال دمج تقنيات الاستشعار اللاسلكي، وبيئة **لاب فيو**، ولغة **بايثون**، وتقنيات الحوسبة السحابية

.

وقد رُتبت موضوعات هذا الكتاب بصورة متسلسلة تبدأ بالأساس النظري وتنتهي بالتنفيذ العملي للنظام. فيستعرض **الفصل الأول** المفاهيم الأساسية لشبكات الحاسب التي تُعد الأساس الذي تقوم عليه أنظمة الاتصال الحديثة. ثم ينتقل **الفصل الثاني** إلى دراسة شبكات الاستشعار اللاسلكية، موضحًا بنيتها، وآلية عملها، وخصائصها، وأهم تطبيقاتها في مجال المراقبة البيئية. ويعرض **الفصل الثالث** بيئة التطوير **لاب فيو** وآلية ربطها ببوابة شبكات الاستشعار اللاسلكية المستخدمة لاستقبال بيانات الحساسات

.

أما **الفصل الرابع** فيتناول صياغة مشكلة البحث من خلال تحليل ظاهرة حرائق الغابات، ودراسة العوامل المؤثرة في حدوثها، وتحديد المتغيرات البيئية المستخدمة في عملية الكشف، بالإضافة إلى توضيح مستويات خطورة الحريق الخمسة وآلية تقييمها اعتمادًا على القيم المرجعية المناسبة لكل متغير. ويختص **الفصل الخامس** بعرض التنفيذ العملي للنظام، بدايةً من استقبال بيانات الحساسات باستخدام **لاب فيو**، ثم نقلها إلى برنامج **بايثون** عبر بروتوكول **تي سي بي** لمعالجة البيانات وتحديد مستوى الخطورة، وصولًا إلى رفع النتائج إلى قاعدة بيانات **فايربيز** وإرسال التنبيهات عند الحاجة. ويختتم **الفصل السادس** الكتاب بعرض أهم النتائج التي تم التوصل إليها، إلى جانب استخلاص الاستنتاجات العامة واقتراح عدد من الاتجاهات المستقبلية التي يمكن أن تسهم في تطوير النظام وزيادة كفاءته

.

ويُؤمل أن يمثل هذا العمل مرجعًا علميًا وعمليًا يفيد الطلاب والباحثين والمهتمين بمجال شبكات الاستشعار اللاسلكية، وأن يوضح كيفية توظيف التقنيات الحديثة في بناء نظام متكامل للكشف المبكر عن حرائق الغابات، بما يسهم في الحد من آثارها وتعزيز كفاءة أنظمة المراقبة البيئية.

الشكر والتقدير

بسم الله الرحمن الرحيم

الحمد لله حمدًا كثيراً طيباً مباركا فيه، حمدًا يليق بجلال وجهه وعظيم سلطانه، نحمده سبحانه وتعالى على ما أنعم به علينا من توفيقٍ وعون، وما وهبنا من عزيمةٍ وصبرٍ حتى أتممنا هذا العمل، والصلاة والسلام على سيدنا محمد ﷺ، خير معلمٍ وقدوة، وعلى آله وصحبه أجمعين.

يسرنا أن نتقدم بخالص الشكر وعظيم الامتنان إلى أستاذنا الفاضل **الأستاذ الدكتور/ عبد اللطيف إبراهيم مصطفى**، لما أولاه لهذا المشروع من اهتمام ورعاية، وما قدمه لنا من توجيه علمي سديد، وملاحظات بناءة، ونصحٍ مخلص، كان لها بالغ الأثر في الارتقاء بهذا العمل وإخراجه بالصورة التي بين أيديكم. لقد كانت خبرته العلمية وتوجيهاته المستمرة مصدرًا للإلهام والدافع نحو بذل المزيد من الجهد والحرص على تحقيق أعلى درجات الجودة والدقة.

كما نتقدم بجزيل الشكر والتقدير إلى السادة أعضاء هيئة التدريس بقسم الاتصالات والإلكترونيات، بكلية الهندسة، جامعة الأزهر، على ما بذلوه من جهود مخلصة في تعليمنا وتأهيلنا، وما غرسوه فينا من قيم البحث العلمي والانضباط الأكاديمي، فكان لعلمهم وتوجيههم الأثر البالغ في مسيرتنا العلمية وفي إنجاز هذا المشروع.

ولا يفوتنا أن نعبر عن اعتزازنا بجميع أعضاء فريق العمل، الذين جسدوا أسمى معاني التعاون والمسؤولية والعمل بروح الفريق الواحد. فقد كان تبادل الأفكار، وتكامل الجهود، والإصرار على تجاوز التحديات، عوامل أساسية أسهمت في الوصول إلى هذا الإنجاز، وعكست قيمة العمل الجماعي في تحقيق الأهداف المشتركة.

كما نتوجه بخالص الامتنان إلى أسرنا الكريمة، تقديرًا لما أحاطونا به من دعم وتشجيع وصبر طوال سنوات الدراسة، وما وفروا لنا من بيئةٍ محفزة كان لها أعظم الأثر في مواصلة مسيرتنا الأكاديمية حتى بلوغ هذه المرحلة.

وفي الختام، نتقدم بالشكر والعرفان إلى كل من أسهم، بقليلٍ أو كثير، بصورة مباشرة أو غير مباشرة، في إنجاز هذا المشروع، سائلين الله تعالى أن يجعل هذا العمل نافعًا، وأن يكتب لنا فيه القبول والتوفيق، وأن يكون لبنةً نافعة في خدمة العلم والمجتمع.

جامعة الأزهر

كلية الهندسة

قسم الهندسة الكهربائية

(إتصالأت وإلكترونيات)

مشروع التخرج

للدرجة العلمية البكالوريوس

تطبيق على شبكة مؤقتة ذاتية التنظيم باستخدام شبكات الاستشعار اللاسلكية

تحت إشراف :

أ.د.م / عبداللطيف إبراهيم مصطفى

مقدم من :

عمر محمد محمد عبد الحميد

أحمد محمد طه محمد

عبدالرحمن شعبان سيد سيد

عبدالرحمن علي عيد سيد

علي إبراهيم محمد علي

محمود محمد السيد

يوليو 2026